

PPPPPPP	RRRRRRR	II	MM	MM	000000
PP PP	RR RR	II	MMMM	MMMM	00 00
PP PP	RR RR	II	MM	MMM MM	00 00
PP PP	RR RR	II	MM	M MM	00 00
PPPPPPP	RRRRRRR	II	MM	MM	00 00
PP	RR RR	II	MM	MM	00 00
PP	RR RR	II	MM	MM	00 00
PP	RR RR	II	MM	MM	000000

# BELSŐ LEÍRÁS

Verzió: 84.1, Kiadva: 84.09.27

Összeállította: Détári György

TARTALOMJEGYZÉK

1. Bevezető. . . . .	4
2. A PRIMO konfiguráció elemei . . . . .	5
2.1 Alapgép. . . . .	5
2.2 Tápegység. . . . .	6
2.3 Klaviatúra . . . . .	6
2.4 Képernyő . . . . .	7
2.5 Hanggenerátor. . . . .	8
2.6 Kazettás magnó . . . . .	9
2.7 Nyomtató . . . . .	9
2.8 Joystick . . . . .	10
2.9 V24 interface. . . . .	10
2.10 COMMODORE BUS . . . . .	11
3. BASIC nyelvleírás . . . . .	12
3.1 Funkcionális billentyűk. . . . .	12
3.2 BASIC változók . . . . .	13
3.3 BASIC konstansok . . . . .	16
3.4 Műveleti jelek, precedencia. . . . .	17
3.5 Automatikus típuskonverziók. . . . .	18
3.6 Programbevitel . . . . .	19
3.7 BASIC parancsok. . . . .	20
3.8 BASIC utasítások . . . . .	27
3.8.1 Általános célú utasítások. . . . .	27
3.8.2 Adatátviteli utasítások. . . . .	34

3.8.3	Kazettakezelő utasítások . . . . .	39
3.8.4	String függvények. . . . .	41
3.8.5	Matematikai függvények . . . . .	43
3.8.6	Speciális függvények . . . . .	48
3.8.7	Hanggenerátor kezelés. . . . .	51
3.8.8	Grafikus függvények. . . . .	51
3.8.9	Assembler rutinok hívása . . . . .	53
4.	Memória kiosztás. . . . .	55
4.1	DCB - Device Control Block . . . . .	55
4.2	Egyéb SYSRAM címek . . . . .	59
4.3	BASRAM címek . . . . .	61
5.	Input/Output. . . . .	63
6.	Karaktergenerátor vezérlés. . . . .	66
7.	Kazettaformátum . . . . .	68
8.	Assemblerből hívható rutinok. . . . .	70
8.1	Display kezelés. . . . .	70
8.2	Printer kezelés. . . . .	74
8.3	Ernyőpuffer kezelés. . . . .	75
8.4	Kazetta kezelő rutinok . . . . .	76
8.5	Egyéb rutinok. . . . .	81
A.	melléklet - klaviatúra fizikai címek. . . . .	86
B.	melléklet - Karakterkészlet . . . . .	87
C.	melléklet - Ernyővezérlő kódok. . . . .	88
D.	melléklet - BASIC hibajelzések. . . . .	89
INDEX.	. . . . .	90

## 1. BEVEZETŐ

A PRIMO mikroszámítógép olcsóságánál fogva a házi számítógépek kategóriájába tartozik, de flexibilitása sok más célra is alkalmassá teszi. Ennek a leírásnak az a célja, hogy BASIC, Assembler vagy vegyes nyelvű programok írásához biztosítsa a szükséges információkat. A könyv elsősorban kézikönyvként szolgál, ami azt jelenti, hogy egy-egy parancs, utasítás vagy terület leírásánál igyekszik teljes információt adni ahelyett, hogy más utasításokra vagy fejezetekre utalna.

A könyvben sem az assembler, sem a BASIC nyelv definíciója nem szerepel. A BASIC nyelv leírásánál a "nyelvjárás" pontos megadása, nem pedig a BASIC programozás technikájának ismertetése volt a fő szempont.



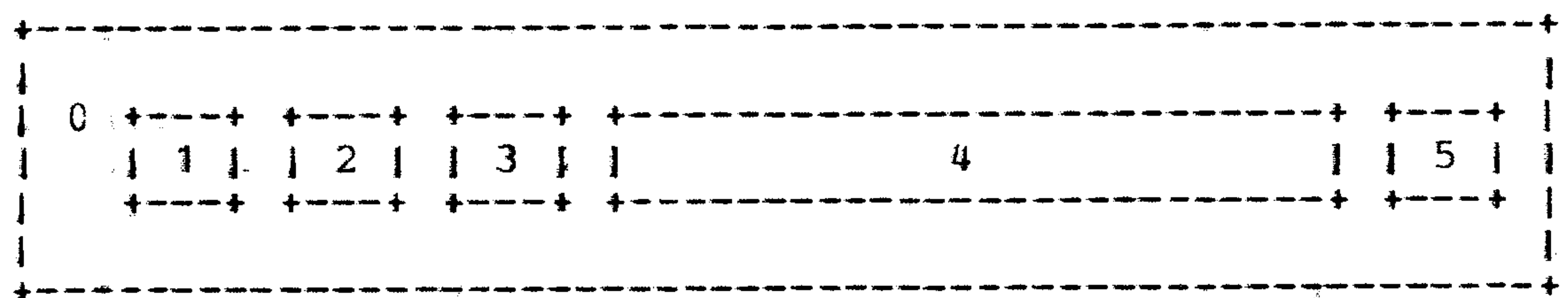
## 2. A PRIMO KONFIGURÁCIÓ ELEMEI

### 2.1 Alapgép

Az alapgép az NDK gyártmányú U880 processzorral működik, amely a ZILOG cég Z80 processzorával kompatibilis. A működőtető órajel frekvenciája 2.5 MHz, de jumperrel 3.75 MHz-ra köthető. A tapasztalat szerint minden U880 processzor képes ezzel a magasabb sebességgel működni, de mivel a gyártó cég specifikációja 2.5, ezért a kikerülő gépek erre a frekvenciára vannak beállítva.

A beépített memória 16 kbyte-os blokkokban bővithető. Az alapkiépítés az A32, amelyben 16 kbyte ROM és 16 kbyte dinamikus RAM található. Az A48 és A64 típusok az alapgéptől a RAM méretében térnek csak el. A ROM egy monitort (4 kbyte) és egy 12 kbyte-os BASIC interpretet tartalmaz. A hátul elhelyezett BUS csatlakozóra fel lehet dugni egy olyan dobozt (puttony), amely a ROM-ot lecseréli pl. egy Assembler programozási rendszerre.

Az alapgép hátoldalán találhatók a csatlakozók a kiegészítő berendezésekhez. A gép hátoldala (hátról nézve):



Az ábra bal felső sarkában a RESET gomb látható. A csatlakozók típusa és szerepe:

- 1 - Kazettacsatlakozó tuchel (TAPE)

- 2 - COMMODORE BUS csatlakozó tuchel
- 3 - Joystick csatlakozó tuchel
- 4 - Z80 BUS kártyacsatlakozó
- 5 - Táp és TV csatlakozó speciális tuchel

## 2.2 Tápegység

Ez az egység a szükséges három tápfeszültségen kívül a TV csatlakozást is biztosítja. A PS A-01 típus TV modulátort tartalmaz az antennán át való csatlakozáshoz, míg a PS A-02 csak a videojel kicsatolását végzi. A tápcsatlakozó bekötése:

- 1 - Földelés
- 2 - Video kimenet (1 Volt PP, 75 Ohm)
- 3 - +12 Volt
- 4 - -5 Volt
- 5 - +5 Volt

## 2.3 Klaviatúra

A klaviatúra lényegében kapacitív érzékelős, és 59 billentyűt tartalmaz. Ezeken szerepel a teljes magyar ABC kis és nagybetűkkel (néhány kivétellel), a számok, speciális jelek, és néhány vezérlőbillentyű (BRK, CLS, RETURN, SHIFT, UPPER, nyilak négy irányban, SPACE)

A klaviatúrán kívül még egy mikrokapcsolót is tartalmaz a



gép, ez a hátlaőon elhelyezett RESET nyomógomb. A klaviatúra és a RESET kódolását software végzi. Az egyes gombok címeit és kezelését lásd az Input/Output fejezetben, és az A. mellékletben.

## 2.4 Képernyő

A PRIMO-hoz bármilyen típusú színes vagy fekete-fehér TV készülék csatlakoztatható, akár az antennacsatlakozón át (36-40 csatorna) a tápegységbe épített modulátor segítségével, akár közvetlen videojel csatlakozással. A PRIMO a képernyőre kétféle fényerejű (sötét/világos) pontokat tud írni. A képernyő mérete egy jumpertől függően 256\*256 vagy 192\*256 lehet. A jelenlegi software csak az utóbbit támogatja. A képernyő frissítését autonóm hardware végzi a memória utolsó 6 illetve 8 kbyte-jából a jumpertől függően. A karaktereket a software bontja pontokra. A karakterkészlet az ASCII szabvány módosított (magyar betűket tartalmazó) változata. Bár a software karaktergenerálás lassúbb, mint a hardware, ugyanakkor sokkal rugalmasabb is, számos képernyőüzem módot megenged. Az üzemmódváltásokat a kiírandó szövegben elhelyezett vezérlőkarakterek adják meg. A karakteres képernyőméret 16 sor \* 42 pozíció alapértelmezésben, míg ha átváltjuk az írás irányát függőlegesre, akkor 21 sor \* 32 pozíció lesz a méret (de a TV készüléket az oldalára kell fektetnünk). A képernyő alapszine lehet világos, ilyenkor a betűk sötéttel íródnak, de válthatjuk a karakterek alapszínét külön is. Mód van a szövegek aláhúzására, alsó és felső

index használatára is. A funkciók vegyesen is használhatóak. Lehetőség van kompozit karakterek létrehozására, mert alaptelmezésben a kiírás előtt a karakterpozíció nem törlődik. A vezérlőkaraktereket a C. melléklet tartalmazza. A software lehetővé teszi a 128 feletti karakterkódok képeinek összeállítását, így tetszőleges új karaktereket definiálhatunk. Az így definiált karakterek nem feltétlenül egy sorosak, azaz pl. definiálhatunk 3-4 soros integráljelet is. A képernyő pontjai egyedileg gyűjthetők, olthatók, így egészen jó minőségű grafikus szolgáltatást kapunk. (ld. képernyőkezelés, grafikus függvények)

## 2.5 Hanggenerátor

A PRIMO-ba be van építve egy kisteljesítményű hangszóró, amelynek kapcsaira programból lehet egyenfeszültséget kapcsolgatni. Így egyszerű négyszőghullám generálására van módunk. A hanggenerátort a software a klaviatúra visszajelzésre és a kazettára írás indikálására használja. Programozását ld. az Input/Output fejezetben.



## 2.6 Kazettás magnó

A kazetta a PRIMO háttértára, ide lehet programokat, képernyőtartalmakat és adatokat menteni. Az információ a kazettára három egyenáramú szint programból történő változtatásával írható fel (ld. Input/Output fejezet). A software a speciális PRIMO struktúrájú file-ok írására és olvasására biztosít támogatást (ld. kazettakezelés, kazettastruktúra), de az interface fizikai kezelésével bármilyen három szintet tartalmazó kazetta olvasható illetve írható. A kazettacsatlakozó DIN szabványos, és a készülék hátoldalán található. Bekötése:

- 1 - Kimenő jel (80 mV, 200 Ohm)
- 2 - Földelés
- 3 - Bejövő jel (1 mV, 4300 Ohm)
- 4 - Magnó vezérlés (+5 V)
- 5 - Magnó tápvezérlés (+5 V)

## 2.7 Nyomtató

A PRIMO-hoz lényegében bármilyen típusú nyomtató kapcsolható, de a legalkalmasabbak azok, amelyek tartalmazzák a svéd karakterkészletet (ilyen pl. a TERTA gyár COMPUTERTA típusa). A csatlakoztatás a hátoldalon lévő BUS csatlakozóra dugható illesztőkártyával történik.

## 2.8 Joystick

Ez az eszköz botkormányhoz hasonlít, amelyen egy külön gomb is található. Rendszerint játékprogramok alakzatait irányíthatjuk segítségével, ilyenkor a külön gomb pl. a tüzelés. A joystick-be öt mikrokapcsoló van beépítve, egy a külön gombnak, négy pedig a különböző irányoknak. A botkormány elmozdítása az iránynak megfelelően egy vagy két mikrokapcsolót zár, így összesen nyolc irányt érzékelhetünk. A PRIMO -hoz két joystick kapcsolható. Az eszköz kezelését ld. az Input/Output fejezetben. A csatlakoztatás a hátlapon található DIN szabványos tuchel-en át történik. Bekötése:

- 1 - 1. joystick bejövő jel
- 2 - Földelés
- 3 - Joystick számláló léptető órajel
- 4 - 2. joystick bejövő jel
- 5 - Tápfeszültség (+5 V)

## 2.9 V24 interface

A magnetofon csatlakozó jumperrel átköthető, és ekkor a V24 szabványfelületnek megfelelő  $\pm 5$  Volt feszültséget adja ki, így módunk van pl. távközlési vonal (MODEM) vezérlésére.

## 2.10 COMMODORE BUS

A hardware-ben minden elő van készítve COMMODORE perifériák, elsősorban floppydisk kezelésére. Az interface programozását ld. az Input/Output fejezetben. A szükséges jelek a hátlaon található DIN szabványos tuchel csatlakozón át adhatók ki. A csatolás Tri-state. A csatlakozó bekötése:

- 1 - Serv. Req
- 2 - Attn
- 3 - Serial data
- 4 - földelés
- 5 - Serial Clock



### 3. BASIC NYELVLEÍRÁS

#### 3.1 Funkcionális billentyűk

CTR	Speciális kódok bevitelére (1-30) szolgál. Csak a kis/nagybetű billentyűkkel együtt nyomva használható. (Ld. B melléklet)
SHIFT	A kalviatúra felső állású jeleit aktiválja. Néhány egyéb karakterre is hat, ld. B melléklet.
UPPER	Megnyomása esetén a begépelte betűk nagybetűállásban kerülnek a gépbe. A funkciót a gomb újabb megnyomásával törölhetjük
RETURN	A sor végét jelzi bevitelnél (Ld. még EDIT parancs és programbevitel).
CLS	Ernyőtörlés
BRK	BREAK - A program, listázás ill. javítás befejezésére szolgál.
RESET	Feltétlen program ill. funkciómegállításra szolgál. A gomb a PRIMO hátlapján található.
<--	Visszalépés karaktertörléssel
SHIFT <--	Sortörlés
:	BASIC utasításszeparátor
.	A legutolsó aktiv (futás, listázás, javítás) sor sorszámát jelenti
?	A BASIC PRINT utasítás rövidítése
'	A BASIC REM utasítás rövidítése

### 3.2 BASIC változók

A BASIC változók neve az angol ABC betűivel kezdődő, és ugyanezen betűkkel vagy számokkal folytatódó karaktersorozat lehet. A nevekre fontos megkötés, hogy nem tartalmazhatnak BASIC alapszavakat, illetve nem egyezhetnek meg velük. Hosszú nevek használata nem célszerű egyrészt a nagyobb helyfoglalás miatt, másrészt mert a BASIC végrehajtás közben a változóneveket két karakterre csonkolja. Erre külön vigyázni kell, különben különböző neveken ugyanazt a változót használhatjuk hibajelzés nélkül. A változókat első előfordulásuk deklarálja. Kezdő értékük mindig nulla, illetve stringváltozókra üres string. A változók típusa négy féle lehet. A típust vagy postfix-el (a név után írt jelzőkarakterrel) vagy a kezdőbetű szerinti típusdeklarációval (DEF... ld. később) adhatjuk meg. Típusdeklaráció és postfix nélkül a változó rövid lebegőpontos típusú lesz. A különböző típusú, de azonos nevű változókat a BASIC megkülönbözteti. Az egyes típusok ábrázolása:

**INTEGER** Postfix jele a %, értéke -32768 -tól +32767 -ig terjedhet. Két byte-on, kettes komplementben ábrázolódik: LSB, MSB, azaz elől van az alacsony helyiérték. A logikai műveletek mindig integerek (egészek) között zajlanak, bitenkénti művelet formájában. A 0 érték a FALSE. A BASIC bármilyen nem nulla értéket TRUE-nak tekint. A reláció műveletek -1 -et (X'FFFF') használnak TRUE értéknek.

**SINGLE** Postfix jele a !, értéke a  $\pm 1.7E\pm 38$  tartományban



mozoghat, max. 7 decimális jegy pontossággal. Deklaráció és postfix nélkül ez a típus az alapértelmezés. Négy byte-on ábrázolódik: LSB, M, MSB, EXP azaz elől van az alacsony helyiérték. Az M a közbülső byte-ot, az EXP a kitevőt jelenti. A szám alakja:  $\text{mantissza} * 2^{**} \text{kitevő}$ . Az  $\text{EXP} = \text{kitevő} + 128$ . A mantissza abszolút értékben ábrázolódik, az MSB legelső bitje az előjel. A számok mindig normalizálva vannak, így az első adatbit mindig 1, ezért ez nem is tárolódik. Az MSB első bitje az előjel után már a mantissza második bitje. A 0-t tiszta nulla jelzi.

DOUBLE Postfix jele a #, értéke a  $\pm 1.7E\pm 38$  tartományban mozoghat, max. 16 decimális jegy pontossággal. Nyolc byte-on ábrázolódik: LSB, M, M, M, M, M, MSB, EXP azaz elől van az alacsony helyiérték. Az M a közbülső byte-ot, az EXP a kitevőt jelenti. A szám alakja:  $\text{mantissza} * 2^{**} \text{kitevő}$ . Az  $\text{EXP} = \text{kitevő} + 128$ . A mantissza abszolút értékben ábrázolódik, az MSB legelső bitje az előjel. A számok mindig normalizálva vannak, így az első adatbit mindig 1, ezért ez nem is tárolódik. Az MSB első bitje az előjel után már a mantissza második bitje. A 0-t tiszta nulla jelzi.

STRING Postfix jele a \$, értéke 0-255 karakter lehet. Három byte-on ábrázolódik: HOSSZ, CIM/LSB, CIM/MSB. A tényleges string a tár tetején allokált külön területen van (ld. Memóriakiosztás és CLEAR parancs)



A változók leírása értékükkel együtt tárolódik a táblázatokban. A VARPTR speciális függvény (ld. ott) mindig a leírás után, az értékre mutat.

A változók más szempontból is csoportosíthatók. Ábrázolásuk:

**SKALAR** A leírás három byte-ból áll: TIPUS, NÉV/2., NÉV/1.  
A típus egyúttal az értékmező hossza is, azaz 2-INTEGER, 3-STRING, 4-SINGLE, 8-DOUBLE.

**TÖMB** Deklarációja kétféleképpen történhet; expliciten a DIM paranccsal, illetve impliciten, úgy hogy tömbként hivatkozunk rá. Az utóbbi esetben az egyes dimenziók a 0-10 tartományban deklarálódnak. A tömbméretet csak egyszer lehet deklarálni. A BASIC megkülönbözteti az azonos nevű tömböt és skalárt. A leírás változó hosszúságú, és a skalárisnál leírt három byte-tal kezdődik. Ezt követi két byte-on (LSB/MSB) a leíró folytatásának hossza. A következő byte a dimenziók száma, majd ezt a dimenziók leírása követi. Minden dimenzióleírás két byte (LSB/MSB), és a dimenzióhatár+1-et tartalmazza. A leírás után jönnek az elemek értékei a skalárisal megegyező formában, sorfolytonosan (azaz a legutolsó index nő a leggyorsabban).

### 3.3 BASIC konstansok

+nnnn        Egész, valós, dupla konstans is megadható így

+nn.nn       Valós és dupla konstans írható le így

+nn.nnE+nn   Valós konstans megadása

+nn.nnD+nn   Dupla konstans megadása

"szöveg"     String konstans megadása.    DATA utasításnál,  
              illette bevitelnél a " elhagyható, ha a konstans  
              elején nincs blank, vagy nem tartalmaz vesszőt.  
              Üres string konstans is megengedett.

A konstansok tárolása a megadás formájától függ. A string-konstanst egyértelműen meghatározza az öt körülvevő kettőspoztróf-pár. A numerikus konstansokra a BASIC a következő vizsgálatst végzi el:

- A konstans duplapontos alakban tárolódik, ha
  - a szám nyolcnál több jegyű
  - normál alakos megadásnál az exponenst D betű jelzi benne
- A konstans rövid lebegőpontos alakban tárolódik, ha a szám nem duplapontos, és
  - tizedespontot tartalmaz
  - normál alakos megadásnál az exponenst E jelzi benne
  - értéke kivülesik a -32768,+32767 tartományon
- Ha a szám nem duplapontos, vagy rövid lebegőpontos, akkor egészként tárolódik.

### 3.4 Műveleti jelek, precedencia

- + Számokra összeadás, stringekre konkatenáció (összefűzés) művelet.
- Kivonás, vagy előjelváltás (unáris mínusz)
- \* Szorzás
- / Osztás
- ↑ Hatványozás (felfele nyíl)

<, >, =, <=, >=, <>

Relációoperátorok. A relációoperátor kétváltozós függvénynek tekinthető, amely egész számot ad eredményül. A függvényérték -1, ha a reláció igaz, és 0, ha nem. A relációkat kifejezésekben és értékadó utasításokban is használhatjuk, pl.

$$I\% = (A <> B) + J\%$$

A <> operátor >< formában is megadható. Megjegyezzük, hogy azonos stringek esetén a rövidebb a kisebb.

AND, OR, NOT Egész számok között végzi bitenkénti ÉS, VAGY illetve tagadás logikai műveleteket. Ha nem egész számok között áll, automatikus konverzió történik.

Két műveleti jel csak abban az esetben állhat egymás után, ha közülük a második az előjelváltást jelző mínusz jel. Az egyes műveletek prioritása csökkenő sorrendben:

- ↑ Hatványozás (felfele nyíl)
- Előjelváltás (unáris mínusz)
- \*, / Szorzás, osztás



$+, -$  Összeadás, konkatenálás, kivonás

$<, >, =, <=, >=, <>$  Relációk

NOT Bitenkénti tagadás

AND Logikai és

OR Logikai VAGY

Az azonos prioritású műveletek között a balról-jobbra szabály dönt. A prioritást tetszőleges mélységű zárójelezéssel változtathatjuk meg.

### 3.5 Automatikus tipuskonverziók

Tipuskonverzió csak a numerikus konstansok között történik.

A konverzió a kiadott műveletektől függ:

$+, -, *$  A műveletek eredményének típusa megegyezik megegyezik a résztvevők közül legmagasabb pontosságú változó típusával

$/$  Osztás esetén a művelet eredményének típus megegyezik megegyezik a résztvevők közül legmagasabb pontosságú érték típusával, azonban egész szám soha sem lehet. Egészek osztása esetén mindkét résztvevő rövid lebegőpontos számmá konvertálódik.

$=, <, >, <=, >=, <>$  Relációműveletek előtt a kisebb pontosságú résztvevő a nagyobb pontosságú résztvevőnek megfelelő típusra konvertálódik.

AND, OR, NOT Logikai műveletek előtt a résztvevő(k) egész típusúvá konvertálódnak. A konverzió miatt túlcsoordulás lehetséges.

### 3.6 Programbevitel

Ha a gép alaphelyzetében egy sort úgy írunk be, hogy számmal kezdődik, akkor a sor programként tárolódik. A sorszám 0-65529 lehet. Ha a sor nem számmal kezdődik, akkor azt a BASIC azonnal végrehajtja. A leghosszabb program/adatsor legfeljebb 5 képernyősor (210 karakter) lehet. A BASIC a stringek belsejét kivéve a szóközöket szűri végrehajtáskor, így, ha a program olvashatóságát feláldozva a sorokat felesleges szóközök nélkül gépeljük, jelentős helymegtakarítást érhetünk el.

A PRIMO BASIC bevitel közben nem elemzi szintaktikusan a szöveget, ezért egy program hibátlanságáról csak akkor lehetünk meggyőződve, ha annak minden utasítását legalább egyszer végrehajtottuk. A kulcsszavak és függvények neveit a BASIC viszont már beolvasás közben felismeri, és egyetlen byte-ként tárolja a helyfoglalás csökkentése céljából. A kódbyte-ok értéke nagyobb 128-nál. Mivel a ¶ (lefele nyíl) bevitelkor speciális SHIFT gombként a kódértékeket X'40' -el növeli, így az alapszavak egy karakterként is beírhatók.

### 3.7 BASIC parancsok

```
+-----+
|      |
| AUTO  |
|      |
+-----+
```

AUTO <nn<,nn>>

Automatikus programsorszámozást tesz lehetővé a beíráshoz. Az első szám a kezdősorszám, a második a lépésköz, mindkettő alapértelmezése 10. A funkcióból a BRK gombbal léphetünk ki. Ha már létező sorszámhoz érünk, a sorszám után \* íródik (pl. 1010\*). A RETURN gomb megnyomása a sort változatlanul hagyja. A RETURN a még nem létező sorokra kihagyott sorszámokat jelent. A CLS gomb törli az ernyőt, de a BASIC még mindig az előzőleg kiírt sor megadását várja.

```
+-----+
|      |
| CLEAR |
|      |
+-----+
```

CLEAR <nn>

A parancs hatására valamennyi változó elveszti korábbi értékét. Az nn paraméter a string terület (ld. Memóriakiosztás) méretét írja elő byte-ban. Ha nem adjuk meg, a méret változatlan marad. A BASIC indulásakor a string terület 50 byte hosszúra áll be. A parancs programból is hívható.



```

+-----+
|       |
|  LOAD  |
|       |
+-----+

```

LOAD <"filenév">>

Hatására a kazettáról az adott nevű illetve név hiányában a következő nem adat file betöltődik a memóriába. A keresés közben átlépett file-ok nevei kiíródnak a képernyőre SKIP: név formában. A file tartalmazhat BASIC vagy Assembler programot, illetve elmentett ernyőképet. Assembler program betöltése előtt a szükséges tárolóterületet a BASIC-tól el kell venni (ld. Memóriakiosztás - BASRAM leírás). A megfelelő BASRAM változót a parancsként kiadott POKE utasítással módosíthatjuk. A névben szerepelhet speciális karakter is. A BASIC a kis és nagybetűket különbözőnek tekinti. A töltésről az ernyő utolsó sorában statisztika jelenik meg:

blckkszám hibaszám FOUND: név

A számok tízes rendszerben íródnak ki két jegyre. A funkció csak a RESET gombbal állítható meg. A parancs kiadható BASIC programból is. Ha assembler programot töltünk, és az nem indul automatikusan, a BASIC program visszkapja a vezérlést. Másik BASIC -et töltve a hívó program törlődik összes változójával együtt, és a betöltött program kap vezérlést. A két program fenntartott memóriaterületen át POKE és PEEK utasításokkal adhat át adatokat egymásnak. A területet a BASIC-tól el kell venni a megfelelő BASIC belső változó módosításával (ld. Memóriakiosztás - BASRAM leírás).

```

+-----+
|      |
|  TEST  |
|      |
+-----+

```

TEST <"név">>

Hatására az adott nevű, illetve név hiányában a soron következő kazettafile olvashatóságát ellenőrzi. Az eredményről az ernyő utolsó sorában statisztika jelenik meg:

blokkszám hibaszám FOUND: név

A számok tízes rendszerben íródnak ki két jegyre. A funkció csak a RESET gombbal állítható meg.

```

+-----+
|      |
|  SAVE  |
|      |
+-----+

```

SAVE "filenév">

Hatására a memóriában levő BASIC program az adott néven kazettára mentődik. A funkció csak a RESET gombbal állítható meg. A filenév speciális jeleket is tartalmazhat. A BASIC a kis és nagybetűket különbözőnek tekinti.

SAVE SCREEN "filenév">

Hatására a képernyő pillanatnyi tartalma az adott néven kazettára mentődik. A funkció csak a RESET gombbal állítható meg. A filenév speciális jeleket is tartalmazhat. A BASIC a kis és nagybetűket különbözőnek tekinti.

```

+-----+
|       |
|  CONT  |
|       |
+-----+

```

## CONT

Hatására a BRK gombbal, END vagy STOP utasítással leállított BASIC program folytatja működését, ha ez lehetséges. A folytatást a program bármilyen módosítása letiltja, azonban a programot listázni, a változókat kiíratni és/vagy módosítani szabad. A megszakított EDIT illetve LIST funkció nem folytatható.

```

+-----+
|       |
| DELETE |
|       |
+-----+

```

## DELETE sorszám

sorszám - sorszám  
- sorszám

Hatására a kijelölt programsorok törlődnek. Az utolsó parancsforma a program elejétől töröl. A megadott sorszámoknak léteznie kell, különben a parancs nem hajtódik végre. Sorszám helyett adható a . (kurrens sor) szimbólum is.

```

+-----+
|       |
|  EDIT  |
|       |
+-----+

```

## EDIT sorszám

A megadott sorszámú BASIC programsort javíthatjuk ezzel a funkcióval. Sorszám helyett adható a . (kurrens sor) szimbólum is. Hiba esetén, ha a BASIC a sort visszairja, azonnal



EDIT funkcióba kerülünk. Ha módosítjuk a javított sor sorszámát, az eredeti sor változatlan marad, és a javított sor az új sorszámmal kerül a programba. Ha az új sorszám már létezik, felülírás történik. Az EDIT -et a következő gombokkal vezérelhetjük:

RETURN      Az EDIT végét jelzi. Ha az editálást RETURN -al kezdjük, a sor nem módosul. Megkezdett EDIT esetén a sor hátralevő része törlődik.

CLS          Ernyőtörlés, de a sor javítása folytatódik.

BRK          BREAK - Az EDIT funkció törlése, a sor nem módosul.

<--          Visszalépés karaktertörléssel

-->          Egy karakter átmásolása az eredeti sorból

↓            a lefele nyíl gomb az EDIT újratekészt jelenti az addigi javítások törlésével

SHIFT <--   az EDIT újratekészt, az addigi javítások maradnak

SHIFT -->   kilépés EDIT-ból a maradék sor átmásolásával

Egyéb        A javított sor bővítése a beírt karakterekkel

```
+-----+
|       |
|  LIST  |
|       |
+-----+
```

LIST <sorszám>

<sorszám - sorszám>

< - sorszám>

<sorszám - >

Hatására a kijelölt programsorok kiíródnak az ernyőre. Sorszám nélkül megadva a funkció a teljes programot kiírja.

Sorszám helyett adható a . (kurrens sor) szimbólum is. A listázás csak addig folyik, míg valamilyen billentyűt nyomva tartunk. A BRK gomb a funkciót félbeszakítja. Ha egy sor listázásánál a megadott sorszám nem létezik, azonnal Ok üzenetet kapunk. Amennyiben határokat írunk elő, és a sorszámok nem léteznek, akkor az alsó határ helyett az azt követő első legnagyobb, felső határ helyett az azt megelőző legkisebb sorszámot veszi a parancs figyelembe. Üres tartomány listázására azonnali Ok a válasz.

```

+-----+
|       |
|  LLIST  |
|       |
+-----+

```

LLIST <sorszám>

<sorszám - sorszám>

< - sorszám>

<sorszám - >

Hatására a kijelölt BASIC programsorok kiíródnak a nyomtatóra. Sorszám nélkül megadva a funkció a teljes programot kiírja. Sorszám helyett adható a . (kurrens sor) szimbólum is. Ha egy sor listázásánál a megadott sorszám nem létezik, azonnal Ok üzenetet kapunk. Amennyiben határokat írunk elő, és a sorszámok nem léteznek, akkor az alsó határ helyett az azt követő első legnagyobb, felső határ helyett az azt megelőző legkisebb sorszámot veszi a parancs figyelembe. Üres tartomány listázására azonnali Ok a válasz. A funkciót csak a RESET gombbal lehet leállítani.

```

+-----+
|       |
|  NEW  |
|       |
+-----+

```

NEW

Hatására a BASIC program teljes egészében törlődik.

```

+-----+
|       |
|  RUN  |
|       |
+-----+

```

RUN <nn>

Hatására a BASIC program elindul a megadott címkétől, illetve ennek hiányában a program legalacsonyabb sorszámú sorától. A RUN parancs törli valamennyi változó értékét, ezért újraindítás céljára a parancsként is kiadható GOTO nn utasítást használhatjuk. Ha a memóriában nincs program, azonnali Ok választ kapunk.

```

+-----+
|       |
|  TRON |
|       |
+-----+

```

TRON

Hatására a BASIC program futás közben érintett sorainak sorszáma kiíródik az ernyőre <címke><címke> ... formában. Programból is kiadható.



```

+-----+
|       |
|  TROFF  |
|       |
+-----+

```

## TROFF

Felfüggeszti a TRON utasítás hatását. Programból is kiadható.

## 3.8 BASIC utasítások

### 3.8.1 Általános célú utasítások

```

+-----+
|       |
|  DEF...  |
|       |
+-----+

```

DEFINT kezdőbetű lista

DEFSNG kezdőbetű lista

DEFDBL kezdőbetű lista

DEFSTR kezdőbetű lista

Hatására azok a változók, amelyeknek kezdőbetűje a listán szerepel, postfix nélkül a megadott típusúak lesznek (INT - egész, SNG - lebegőpontos, DBL - duplapontos, STR - string). A postfix megadása felülbírálja a DEF.. utasítás hatását. A lista állhat egyetlen elemből, vagy elemek vesszővel elválasztott felsorolásából. Egy elem vagy egy betű, vagy egy betűtartomány lehet. A tartományt az ABC rendben leírt betűk közé rakott elválasztójel jelzi, pl. DEFINT A,G-J,T Ha egy változó több DEF... listán szerepel, mindig az utolsó specifikáció lesz érvényes. A változótipusok többször is átdefi-

niálhatók, ilyenkor a korábbi tartalmaik a postfix megadásával érhetők el.

```
+-----+
|         |
|  DIM    |
|         |
+-----+
```

DIM név(n1<,n2<,n3 ...>>>)<,név( .....>

Az utasítás a tömbök dimenziószámának és dimenzióhatárainak leírására szolgál. A név a tömb neve, n1, n2 .. pedig az egyes dimenziók felső határa. Az alsó határ mindig nulla. A DIM utasításban leírt tömbnév a DIM előtt nem szerepelhet a programban. A dimenziók helyén változó is állhat, ha a programfutás során a DIM utasítás előtt már értéket kapott.

```
+-----+
|         |
|  LET    |
|         |
+-----+
```

LET változó = kifejezés

Az utasítás az aritmetikai értékadást jelöli. A LET alapszó elhagyható. A változó helyén állhat skaláris változó vagy tömbelem. A kifejezés változókból, tömbelemekből, konstansokból, függvényekből és a már felsorolt műveleti jelekből állhat. A BASIC a kiértékelés közben automatikus típuskonverziót végez. A kifejezésben a string-ek mindig " jelek közé vannak fogva. A kifejezés a benne szereplő elemek típusa szerint lehet aritmetikai, vagy stringkifejezés. A két kifejezéstípus nem vegyithető, azaz stringek és aritmetikai változók között nincs automatikus típuskonverzió (természetesen aritmetikai kifejezés is tartalmazhat stringeket,

ha azck egy reláció két oldalán állnak).

END

END

Az utasítás a BASIC program futását leállítja. A program fizikai vége egyenértékű egy END utasítással.

A black and white graphic of a stop sign. It consists of a square with a thick border. Inside the square, the word "STOP" is written in a bold, sans-serif font, centered horizontally and vertically. The background of the sign is white, and the border and text are black.

STOP

Az utasítás a BASIC program futását leállítja, hatása azonos a BRK billentyűvel, azaz kiíródik a

Break in nn

üzenet. A program futása a CONT paranccsal folytatható. A STOP utasítást töréspontok elhelyezésére használhatjuk a programbelövés során. Megálláskor a változók tartalma kiiratható és/vagy megváltoztatható.

```

+-----+
|       |
|  GOTO  |
|       |
+-----+

```

GOTO sorszám

Az utasítás feltétel nélküli ugrást okoz a megadott sorszá-  
mú programsorra. Utasításként kiadva újraindíthatjuk a prog-  
ramot anélkül, hogy a BASIC a változóértékeket törölné.



```

+-----+
|       |
|  GOSUB  |
|       |
+-----+

```

### GOSUB sorszám

Az utasítás meghívja a megadott sorszámú programsorral kezdődő szubrutint (azaz feljegyzi a visszatérési címet és ugrik a megfelelő programsorra). Utasításként is kiadható, így hibakeresésnél felhasználhatjuk a program után irt szubrutin hívására a töréspontokban. A szubrutin pl. kiirathatja a fontosabb változók értékét.

```

+-----+
|       |
|  RETURN  |
|       |
+-----+

```

### RETURN

Az utasítás visszatér a GOSUB utasítással hívott szubrutinból. A programfutás a GOSUB-ot követő utasítással folytatódik.

```

+-----+
|       |
|   ON   |
|       |
+-----+

```

### ON változó GOTO sorszám1<,sorszám2 ...>

Az utasítás hatására a BASIC a változó tartalma szerint választ a felsorolt sorszámlistából egy elemet. A változó értékét modulo 256 tekinti. Ha az érték 0, vagy nagyobb, mint a felsorolt sorszámok darabszáma, a következő utasítás hajtódik végre, egyébként feltétel nélküli ugrás történik a megadott sorszámú programsorra.

ON változó GOSUB sorszám1<,sorszám2 ...>

Az utasítás hatására a BASIC a változó tartalma szerint választ a felsorolt sorszámlistából egy elemet. A változó értékét modulo 256 tekinti. Ha az érték 0, vagy nagyobb, mint a felsorolt sorszáмок darabszáma, a következő utasítás hajtódik végre, egyébként meghívja a megadott sorszámú programsorral kezdődő szubrutint (azaz feljegyzi a visszatérési címet és ugrik a megfelelő programsorra).

ON ERROR GOTO sorszám

Az utasítás hatására a BASIC feljegyzi a programsorszámot, és hiba esetén a vezérlést a megadott sorra adja. Ha sorszám helyett 0-t adunk, törli a korábban beállított hibarutin címet.

```
+-----+  
|       |  
| RESUME |  
|       |  
+-----+
```

RESUME <sorszám>

<NEXT>

<0>

Csak akkor adható ki, ha a BASIC az ON ERROR utasítás hatására aktiválta a hibarutint. Az utasítás célja a hibarutinból való visszatérés. Ha a paraméter 0, vagy hiányzik, akkor a BASIC újratekint a hibázó utasítást. Ha a paraméter a NEXT szó, a vérehajtás a hibázó utasítást követő utasításon folytatódik. Ha sorszámot adunk meg, a BASIC a kijelölt sorra ugrik. a RESUME utasítás törli a hibaállapotot.

```

+-----+
|      |
|  ERROR  |
|      |
+-----+

```

### ERROR hibakód

Az utasítás az ON ERROR rutinok belövésére szolgál. A megadott kódú hibát szimulálja. A hibakódok a D. mellékletben találhatók. Ha érvénytelen hibakódot adunk meg, akkor a hiba a 20-as kódú UE (user error) lesz.

```

+-----+
|      |
|  FOR  |
|      |
+-----+

```

FOR változó = kifejezés1 TO kifejezés2

<STEP kifejezés3>

Cikluskezdő definíció. A ciklust a NEXT utasítással kell zárni. A megadott ciklusváltozó kezdeti értéke az első kifejezés, végértéke a második kifejezés, míg a harmadik kifejezés a lépésközt adja meg. Ha a lépésközt nem adjuk meg, akkor a BASIC +1 -et használ. Ha a ciklusváltozó értékét a ciklusban csökkenteni akarjuk (azaz kifejezés1 > kifejezés2), akkor a negatív lépésközt (pl. STEP -1) meg kell adnunk. A ciklus egyszer mindenképpen lefut. Ha a lépésköz nulla, végtelen ciklus alakul ki. A BASIC a végértéket és a lépésközt a ciklus elején menti, így ezek a ciklusmagban megváltoztathatók. Kerüljük azonban a ciklusváltozó átírását, mert ez kiszámíthatatlan mellékhatásokkal járhat. Ciklusból csak akkor lépünk ki, ha oda vissza is térünk.



```

+-----+
|       |
|  NEXT  |
|       |
+-----+

```

NEXT <változó1 <,változó2 ...>>

Ciklusvég definiálása. A ciklusváltozó(k) megadása nem kötelező, de ha megadtuk, egyeznie kell az éppen aktív FOR utasítás változójával. Több ciklus közös végét adhatjuk meg a változók felsorolásával, ilyenkor az elsőként megadott változó a legbelső ciklushoz kell tartozzon. A programfutás során végrehajtott NEXT utasítás, függetlenül az elhelyezésétől, mindig a legutóbb megkezdett ciklushoz fog tartozni.

```

+-----+
|       |
|  IF    |
|       |
+-----+

```

IF egészskifejezés THEN utasítás1

<:ELSE utasítás2>

Feltételes szerkezet definiálása. Az egészskifejezés természetesen lehet reláció is. A relációk értéke 0 (FALSE), vagy -1 (TRUE) lehet. A BASIC minden 0-tól különböző értéket igaznak (TRUE) tekint. Ha a kifejezés igaz, végrehajtódik a THEN után írt utasítás1 rész, amely egy vagy több utasítás lehet. Az igaz ágat a sor vége, vagy az ELSE utasítás határolja be. Ha az utasítás egyetlen ugrás, akkor a THEN után elég a sorszámot megadni. Ebben az esetben az ELSE előtt felesleges a : megadása. Ha az ELSE ág szerepel, ennek hatóköre is a sor vége, és GOTO utasítás esetén itt is elegendő a sorszám megadása. A teljes IF-THEN-ELSE struktúra hossza legfeljebb egy programsor lehet. Egymásba skatulyázott fel-

tételes struktúrák esetén az ELSE mindig a legbelső IF -hez tartozik. Ha ez nem felel meg, üres ELSE ágakat kell beiktatni.

```
+-----+
|       |
|  REM  |
|       |
+-----+
```

REM tetszőleges szöveg

Az utasítás kommentárok írására szolgál, és rövidíthető aposztrófra. Mivel a REM utáni teljes sort a BASIC kommentárnak tekinti, a REM csak az utolsó utasítás lehet egy programsorban.

### 3.8.2 Adatátviteli utasítások

```
+-----+
|       |
| PRINT |
|       |
+-----+
```

PRINT lista<;TAB (n);lista ...>

Az utasítás a képernyőre való kiíratásra szolgál. A lista elemei lehetnek konstansok, változók, tömbelemek és kifejezések. Az elemeket egymástól vagy vesszővel, vagy pontosvesszővel választhatjuk el. A vessző hatására 16-os tabuláció történik, míg a pontosvessző elválasztás nélkül egymás mellé írja az elemeket (de a számok után mindig íródik egy szóköz). Az írás sorfolytonosan történik, azaz, ha a kurrens sor tele van, a következőben folytatódik. Ha az ernyő tele van, az írás mindig az utolsó sorba történik, míg az ernyőtartalom ugyanúgy csúszik felfele, mint az írógépben a



papír (SCROLL). A lista végén soremelés történik. Ha ezt el akarjuk kerülni, a lista végére pontosvesszőt kell írni. Üres string kiírása esetén a kurrens pozíció nem lép tovább. Speciális lehetőség a TAB(x) függvény, amely a következő elem kiíratási pozícióját vezérli. A TAB függvényekben megadott értékek 0-41 között lehetnek (a TAB csak az alsó hat bitet vizsgálja), és egy PRINT utasításon belül monotonon növekedniük kell. A PRINT utasításban ernyővezérlő karakterek is kiadhatók a listára írt CHRE függvény segítségével. Az egész számok kiírása a szokásos formában történik. A valós számok hét jegyig egész, efelett normál alakban íródnak ki. A duplapontos számok 16 jegyig íródnak ki egész formában, és efelett jelennek meg normál alakban. A pozitív előjel helyett szóköz íródik ki.

```
PRINT y,x,lista<;TAB(n);lista ..>
```

Az utasítás megegyezik a PRINT utasítással, azzal a különbséggel, hogy a kiírás az y,x pozíción kezdődik. Az y 0-15, az x 0-41 lehet (illetve függőleges írás esetén 0-20, 0-31, ld. a C. mellékletet).

```
PRINT USING "formátum";lista
```

A parancsnak ez a formája számok formátum szerinti kiíratását teszi lehetővé. A formátumot stringváltozóban is megadhatjuk. A formátum elemei:

- #           A numerikus érték számjegyei. Közéjük keverve csak pont és vessző írható.
- .
- A tizedespont helyét jelöli ki.
- ,
- Ha bárhol előfordul az érték számjegyei közt, a számjegyek hármas csoportokban, vesszővel elválasztva íródnak ki.



- + A mező előtt vagy után állhat. Kijelöli az előjel helyét, és megadja, hogy a pozitív előjel + -ként íródjon ki.
- A mező előtt vagy után állhat. Kijelöli az előjel helyét, és megadja, hogy a pozitív előjel szóközként íródjon ki.
- AAAA A függőleges nyilak a lebegőpontos kitevő helyét adják meg.
- \*\* A mező előtt állhat, és azt jelzi, hogy a vezető nullák helyett csillagok nyomtatódjanak.
- EE A mező előtt állhat, és azt jelzi, hogy a szám elé közvetlenül egy E jel nyomtatódjék. A E a szám és az előjele közé kerül.
- \*\*E A mező előtt állhat, és azt jelzi, hogy a vezető nullák helyett csillagok nyomtatódjanak, a szám elé közvetlenül pedig egy E jel.
- % % Stringmezőt jelöl ki. A listán az ennek megfelelő pozíción stringváltozónak vagy stringnek kell állnia.
- ! A listán az ennek megfelelő pozíción stringváltozónak vagy stringnek kell állnia. A ! hatására ennek első karaktere íródik ki.
- Egyéb Minden más beírt karakter szöveggént kiíródik, és befejezi az egy számhoz tartozó mező leírását.

Ez az összetett utasítás jelentősen megkönnyíti a képernyőn a táblázatok összeállítását, mert így szabályozhatjuk, hogy egy szám a nagyságától függetlenül mindig ugyanannyi pozíciót foglaljon. Egy szám leírása legfeljebb 24 karakter lehet.

Ha a formátum túl hosszú, a maradék figyelmen kívül marad, míg ha lista hosszabb, mint a formátum, akkor a BASIC elől-ről kezdi a formátum értelmezését.

```
+-----+
|       |
|  LPRINT  |
|       |
+-----+
```

`LPRINT lista<;TAB(n);lista ...>`

Az utasítás megegyezik a PRINT utasítással, azzal a különbséggel, hogy a kiírás a nyomtatóra történik, és csak az X'0D', X'0A' - soremelés, X'0C' - lapdobás vezérlőkódok hajthatódnak végre.

`LPRINT USING "formátum";lista`

Az utasítás megegyezik a PRINT USING utasítással, azzal a különbséggel, hogy a kiírás a nyomtatóra történik.

```
+-----+
|       |
|  INPUT  |
|       |
+-----+
```

`INPUT <"szöveg";>lista`

Az utasítás a klaviatúráról olvas konverzióval. Kiadásakor kiíródik a szöveg, majd egy kérdőjel, és megjelenik a cursor. A lista lehet egy vagy többelemű, string és aritmetikai változók egyaránt szerepelhetnek rajta. A válaszadást többféleképpen végezhetjük, vagy begépeljük az összes értéket vesszővel elválasztva, vagy egyesével adjuk meg őket. A bevittelt mindig RETURN-al jelezzük. Ha egy stringváltozóba blank értéket is be akarunk írni, akkor kettősaposztrófok közé zárva kell megadnunk. Ha elemenkénti beadás van, az

utasítás az újabb elemeket két kérdőjellel kéri. Ha hibás adat került be, kiíródik a ?REDO szöveg, és ilyenkor az egész listát előlről be kell írni. Ha egyszerre írtuk be a listát, és túl sok elemet adtunk meg, akkor az ?EXTRA IGNORED üzenet jelenik meg, és a fölösleges elemeket az utasítás figyelmen kívül hagyja.

```
+-----+
|       |
| INKEY  |
|       |
+-----+
```

stringváltozó=INKEY

Ez a függvény egy egykarakteres stringet ad eredményül. Ha a klaviatúrán le van nyomva egy gomb, akkor ennek értékét kapjuk meg, míg ha nincs, akkor az eredmény az üres string. Az utasítás hatására a cursor nem jelenik meg, a beírt karakter pedig nem íródik ki az ernyőre.

```
+-----+
|       |
| DATA  |
|       |
+-----+
```

DATA lista

Az utasítás adatmező leírásra szolgál, amelyet aztán a READ utasítással helyezhetünk el tömbökbe és változókba. A lista elemei lehetnek fix és lebegőpontos számok, valamint stringek is. A stringeket nem kell aposztrófok közé zárni, csak akkor, ha vezető szóközt vagy vesszőt tartalmaznak. A programba bárhol beírt DATA utasításokat a BASIC egyetlen nagy tömbként kezeli.



```

+-----+
|       |
|  READ  |
|       |
+-----+

```

### READ lista

A DATA utasítás(ok)kal leírt adatmező tömbökbe/változókba való elhelyezésére szolgál. Elvégzi a tipusegyeztetést, és a szükséges konverziót. Az olvasást a legelső DATA utasításnál kezdő, majd minden READ az előző READ pozíciójától folytatja. Az olvasás helyét befolyásolhatjuk a RESTORE utasítással. Ha több adatot akarunk olvasni, mint amennyi a DATA mezőben szerepel, OD (Out of Data) hibát kapunk.

```

+-----+
|       |
|  RESTORE  |
|       |
+-----+

```

### RESTORE <sorszám>

Hatására a következő READ utasítás a feldolgozást a megadott sorszámú DATA utasítástól kezdő. Ha a sorszámot nem adtuk meg, a feldolgozás a DATA mező elejétől kezdődik.

### 3.8.3 Kazettakezelő utasítások

```

+-----+
|       |
|  OPEN  |
|       |
+-----+

```

### OPEN string

Az utasítás a megadott string vagy stringváltozónak megfelelő adatfile címkét kikeresi a kazettán. Ha a stringet nem

adjuk meg, a soron következő adatfile-t nyitja meg.

```
+-----+
|       |
| CREATE |
|       |
+-----+
```

#### CREATE string

Az utasítás a megadott string vagy stringváltozónak megfelelő file címkét felírja a kazettára.

```
+-----+
|       |
| PRINT# |
|       |
+-----+
```

#### PRINT# lista

Az utasítás megegyezik a PRINT utasítással, azzal a különbséggel, hogy a kiírás a kazettára történik.

```
+-----+
|       |
| INPUT# |
|       |
+-----+
```

#### INPUT# lista

Az utasítás a kazettáról olvas konverzióval. A lista lehet egy vagy többelemű, string és aritmetikai változók egyaránt szerepelhetnek rajta. A lista végén a BASIC új sor jelet vár, tehát a lista formátumának szigorúan egyeznie kell a kazettaformátummal. Adathiba, vagy túlolvasás esetén a program hibajelzéssel leáll.

### 3.8.4 String függvények

```
+-----+
|       |
|  ASC  |
|       |
+-----+
```

ASC (string)

A függvény a megadott string vagy stringváltozó első karakterének kódértékét adja.

```
+-----+
|       |
| CHR$  |
|       |
+-----+
```

CHR\$ (kifejezés)

A függvény a megadott kifejezés értékének megfelelő kódértékű karaktert adja.

```
+-----+
|       |
|  LEN  |
|       |
+-----+
```

LEN (string)

A függvény a megadott string vagy stringváltozó kurrens hosszát adja.

```
+-----+
|       |
| LEFT$ |
|       |
+-----+
```

LEFT\$ (string,n)

A függvény a megadott string vagy stringváltozó bal szélső n karakterét adja. Az n helyén kifejezés is állhat.



```

+-----+
|       |
|  RIGHTE  |
|       |
+-----+

```

RIGHTE (string,n)

A függvény a megadott string vagy stringváltozó jobb szélső n karakterét adja. Az n helyén kifejezés is állhat.

```

+-----+
|       |
|  MIDE  |
|       |
+-----+

```

MIDE (string,n,m)

A függvény a megadott string vagy stringváltozó n. -dik pozíciójától m karaktert ad. Az n helyén kifejezés is állhat. A pozíciók számozása 1 -től kezdődik.

```

+-----+
|       |
|  STRINGE  |
|       |
+-----+

```

STRINGE (n,m)

Az m helyén kettősaposztrófok között állhat egy karakter, vagy megadható a karakter kódértéke is kifejezés formájában. A függvény az m -el megadott karaktert n -szer ismételve állít elő új stringet.

```

+-----+
|       |
|  STRE  |
|       |
+-----+

```

### STRE (kifejezés)

A függvény a megadott kifejezést a PRINT utasításnak megfelelő formában karakteresre konvertálja. A kapott string a függvény értéke, amely abban különbözik csak a PRINT által nyomtatott képtől, hogy a végén nincs elválasztó blank.

```

+-----+
|       |
|  VAL   |
|       |
+-----+

```

### VAL (string)

A függvény a megadott string első vagy egyetlen numerikus mezejét számmá konvertálja. Ha a stringnek nincs numerikus része, az eredmény nulla lesz.

## 3.8.5 Matematikai függvények

```

+-----+
|       |
|  ABS   |
|       |
+-----+

```

### ABS (x)

A függvény a megadott x kifejezés abszolút értékét adja.

```

+-----+
|       |
|  ATN  |
|       |
+-----+

```

ATN (x)

A függvény a megadott x kifejezés Arcus Tangens -ét adja radiánban.

```

+-----+
|       |
| CDBL  |
|       |
+-----+

```

CDBL (x)

A függvény a megadott x kifejezést dupla pontossággal számolja ki.

```

+-----+
|       |
|  CINT  |
|       |
+-----+

```

CINT (x)

A függvény azt a legnagyobb egészsámot adja, amely a megadott x kifejezésnél kisebb (Azaz  $CINT(-0.5)=-1$ ,  $CINT(0.5)=0$ ). Az eredmény egész.

```

+-----+
|       |
|  COS  |
|       |
+-----+

```

COS (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Cosinus -át adja.



```

+-----+
|       |
|  CSNG  |
|       |
+-----+

```

CSNG (x)

A függvény a megadott x kifejezés értékét rövid lebegőpontos számmá konvertálja.

```

+-----+
|       |
|  EXP   |
|       |
+-----+

```

EXP (x)

A függvény a megadott x kifejezés exponenciális ( $e^x$ ) függvénye.

```

+-----+
|       |
|  FIX   |
|       |
+-----+

```

FIX (x)

A függvény a megadott x kifejezésből törli a tizedespont utáni számjegyeket, és az eredmény az így kapott egészszám. (Azaz  $\text{FIX}(-0.5)=0$ ,  $\text{FIX}(0.5)=0$ )

```

+-----+
|       |
|  INT   |
|       |
+-----+

```

INT (x)

A függvény azt a legnagyobb egészszámot adja, amely a megadott x kifejezésnél kisebb. Az eredmény lebegőpontos. (Azaz  $\text{INT}(-0.5)=-1$ ,  $\text{INT}(0.5)=0$ )

```

+-----+
|       |
|  LCG  |
|       |
+-----+

```

LOG (x)

A függvény a megadott x kifejezés tízesalapú logaritmusát adja.

```

+-----+
|       |
|  LN   |
|       |
+-----+

```

LN (x)

A függvény a megadott x kifejezés természetes (e) alapú logaritmusát adja.

```

+-----+
|       |
|  RND  |
|       |
+-----+

```

RND (x)

A függvény véletlen számokat generál. Ha x helyén 0 szerepel, akkor 0 és 1 közé eső valós számokat kapunk, amelyek közt a két határ nem fordul elő. Ha x helyén pozitív szám áll, akkor a generátor 1 és x közé eső egész számokat generál, amelyek között a két határ előfordulhat.

```

+-----+
|       |
|  RANDOM  |
|       |
+-----+

```

## RANDOM

Ha az RND(X) függvény használatakor a program két különböző futásához különböző értékek szükségesek, a RANDOM függvény segítségével az RND(X) kezdőértéke véletlenszerűen megváltoztatható.

```

+-----+
|       |
|  SGN   |
|       |
+-----+

```

## SGN (x)

Az x kifejezés előjel függvénye, azaz pozitív x-re 1, negatívra -1, nullára 0 egészszámot ad eredményül.

```

+-----+
|       |
|  SIN   |
|       |
+-----+

```

## SIN (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Sinus -át adja.

```

+-----+
|       |
|  SQR   |
|       |
+-----+

```

## SQR (x)

Az x kifejezés négyzetgyökét adja meg.



```

+-----+
|       |
|  TAN  |
|       |
+-----+

```

TAN (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Tangens ét adja.

### 3.8.6 Speciális függvények

```

+-----+
|       |
|  ERL  |
|       |
+-----+

```

ERL

A függvény a gép bekapcsolása óta utoljára hibát okozó sor sorszámát adja meg. Ha hiba még nem volt, 0-t ad vissza, illetve assembler rutinból hívva -32767 -et.

```

+-----+
|       |
|  ERR  |
|       |
+-----+

```

ERR

A függvény a legutoljára elkövetett hiba kódját adja meg. A D. mellékletben felsorolt hibakódokat az

$ERR/2+1$

kifejezésből kaphatjuk meg.

```

+-----+
|       |
|   POS   |
|       |
+-----+

```

POS (0)

A függvény a cursor kurrens pozícióját (0 - 41) adja meg a display kurrens során belül.

```

+-----+
|       |
|   FRE   |
|       |
+-----+

```

FRE (skaláris változó)

A függvény a megadott változó típusától függően a szabad memória, illetve szabad stringmemória méretét adja byte-okban.

```

+-----+
|       |
|   INP   |
|       |
+-----+

```

INP (x)

A függvény az x (0-255) című port-ról beolvas egy byte-ot (ld. Input/Output fejezet).

```

+-----+
|       |
|   OUT   |
|       |
+-----+

```

OUT x,b1<,b2...>

A függvény az x (0-255) című port-ra kiküldi a megadott adatsort (ld. Input/Output fejezet).

```

+-----+
|       |
|  PEEK  |
|       |
+-----+

```

PEEK (x)

A függvény az x című memóriabyte-ot adja eredményül. Ha a cím nagyobb, mint 32767, akkor helyette a 65536-x értéket kell megadni.

```

+-----+
|       |
|  POKE  |
|       |
+-----+

```

POKE x,b1<,b2....>

A függvény az x című memóriabytetől betölti a megadott adatsort. Ha a cím nagyobb, mint 32767, akkor helyette a 65536-x értéket kell megadni.

```

+-----+
|       |
|  VARPTR  |
|       |
+-----+

```

VARPTR (x)

A függvény az x nevű változó vagy tömbelem címét adja meg.



### 3.8.7 Hanggenerátor kezelés



BEEP x,n<;x,n .....>

Ez az utasítás a hanggenerálást segíti. Ellentétben pl. a klaviatúra leütéskor keletkező hangokkal, ez kimegy a magnó-csatlakozóra, így felvehető, illetve erősítőre vezethető.

Az x érték a fél rezgésidőt adja meg a következő módon:

$$T/2 = 8.26 * x + 36.44 \text{ usec}$$

A generált hangok tartománya 400Hz-18kHz lehet. Az n értéke a félhullámok számát, azaz a hang időtartamát adja meg.

### 3.8.8 Grafikus függvények

Megjegyezzük, hogy a képernyő az itt felsorolt parancsok és függvények használata nélkül is kezelhető a PEEK/POKE függvényekkel. A képernyőpuffer a memória utolsó 6 kbyte részét foglalja el, így címe a PRIMO memóriaméretétől függ:

PRIMO A32 26624

PRIMO A48 43008 => -22528

PRIMO A64 59392 => -6144

A képernyő nyolc vízszintesen egymás mellett fekvő pontja felel meg egy byte-nak. A puffer kezdőcíme a képernyő bal felső sarkának felel meg. A puffer byte-jait a PEEK/POKE függvényekkel manipulálva tetszőleges ábra létrehozható. Az 1-es értékű bitek mindig világosak lesznek. Mivel vízszintes

irányban 256 pont van, egy pntsornak 32 byte felel meg. Az  
ernyő 192 pontsort tartalmaz.

```
+-----+  
|       |  
|  SET  |  
|       |  
+-----+
```

SET (x,y)

A parancs a képernyő x,y koordinátájú pontját a képernyő  
alapszínével (sötét/világos) ellentétesre váltja. A (0,0)  
pont a képernyő bal alsó sarka, a címzés nullától indul. Az  
x= 0-255, az y= 0-191 tartományban adható meg.

```
+-----+  
|       |  
| RESET |  
|       |  
+-----+
```

RESET (x,y)

A parancs a képernyő x,y koordinátájú pontját a képernyő  
alapszínével (sötét/világos) egyezőre váltja. A (0,0) pont  
a képernyő bal alsó sarka, a címzés nullától indul. Az x=  
0-255, az y= 0-191 tartományban adható meg.

```
+-----+  
|       |  
| POINT |  
|       |  
+-----+
```

POINT (x,y)

A függvény a képernyő x,y koordinátájú pontját teszteli.  
Ha a pont értéke a képernyő alapszínével (sötét/világos)  
egyezik, a függvényérték 0, ellenkező esetben -1 lesz. A  
(0,0) pont a képernyő bal alsó sarka, a címzés nullától

indul. Az  $x = 0-255$ , az  $y = 0-191$  tartományban adható meg.

```
+-----+
|       |
|   CLS   |
|       |
+-----+
```

CLS

A parancs a képernyőt törli, és a kurrens pozíciót a bal felső sarokba helyezi.

### 3.8.9 Assembler rutinok hívása

```
+-----+
|       |
|   CALL   |
|       |
+-----+
```

CALL (cim <,egész paraméter> )

Ez a funkció a megadott című assembler programot indítja. Ha a cím nagyobb, mint 32767, akkor helyette a 65536-cím értéket kell megadni. Ha az opcionális paramétert megadjuk, akkor az assembler program ezt a DE regiszterpárban kapja meg. \*Assembler szubrutin elhelyezésére többféle lehetőség is adódik.

- A BASIC program tartalmazza az assembler rutint számok formájában. Ebben az esetben írhatjuk a rutint DATA utasítással, és innen tömbbe olvasva, vagy a memória szabad területeire a POKE függvénnyel betöltve használhatjuk. A POKE VARET(tömb(0)),x,x,x... sorozattal közvetlenül is tömbbe helyezhetjük a programot. A tömbben futó programokat önáttelezésre kell írni, mert a BASIC a tömb kezdőcímét minden új skaláris változó felvételekor módosítja. A legegyszerűbb

\* A visszaadandó értéket a HL-be kell rakni. Belépőként  
BC = return address (stackpointer), DE = param, HL = belépési pont



módszer az, ha a programban csak relatív ugró utasítások vannak, és az adatterületet a BASIC tartalmazza. A program hívása a CALL (VARPTR(tömb1(0)),VARPTR(tömb2(0))) utasítással történhet, ahol a tömb1 a programot, a tömb2 az átadott

paramétereket és munkaterületet tartalmaz. A fűzőhívás végén *HL-ben*  
- Az assembler programot vagy kézzel, vagy BASIC-ből kazettáról töltjük.

Az assembler rutinokra vonatkozólag ld. a BASRAM leírást, és a szalagformátum leírást. Megjegyezzük, hogy az assembler rutinok híváskor minimálisan 58 byte hosszú STACK-et kapnak.

Ha a BASIC-től területet veszünk el a programnak, akkor az a LOAD funkcióval betölthető mind a BASIC-ből, mind parancsként kiadva. A nem automatikus indítású program ezután CALL utasítással hívható (parancsként pl. PRINT CALL(cim), vagy I%=CALL(cim) ). A RUN parancs mindig a BASIC programot indítja.

Pontos ! A rendszer az IX regisztert beállítja a DCB címére, ezt nem szabad rontani, ha a rendszerrutinokat használni akarjuk ! Az IY regiszter, és a második regiszterkészlet szabadon használható.

#### 4. MEMÓRIA KIOSZTÁS

X'0'	+-----+	
	MONITOR	4 kbyte MONITOR (SYSTEM) ROM
X'1000'	+-----+	
	BASIC	12 kbyte BASIC ROM
X'4000'	+-----+	
	SYSRAM	SYSTEM RAM terület
X'407F'	+-----+	
	BASRAM	BASIC RAM terület - Belső változók
	+-----+	
	PROG	BASIC RAM terület - BASIC program
	+-----+	
	SCALAR	BASIC RAM terület - BASIC skaláris
		változók
	+-----+	
	ARRAY	BASIC RAM terület - BASIC tömbök
	+-----+	
	--	BASIC RAM terület - Üres hely
	+-----+	
	STACK	BASIC RAM terület - STACK
	+-----+	
	STRING	BASIC RAM terület - STRING memória
		(ld. CLEAR parancs)
X'E800'	+-----+	
	SCREEN	HARDWARE terület - Képernyő puffer.
		32k/16k RAM esetén
		X'A800'/X'6800' a cím. Mérete
		alapértelmezésben 6 kbyte. A cím
		váltható X'C800'/X'8800' -ra.
	+-----+	

##### 4.1 DCB - Device Control Block

A DCB a SYSRAM legfontosabb vezérlőblokkja. Címét az IX regiszter tartalmazza, de a ROM X'1B' címéről is ki lehet olvasni. A mostani verzióban a DCB címe X'4042'. A DCB más helyre még kerülhet, de a felosztása már nem fog változni.

Kiosztása:

- +0.(1) Utolsó klaviatúra karakter.
- +1.(1) Klaviatúra repeat számláló.

+2. (1) Utolsó aktiv gomb címe. Az első két bit: X'80' - UPPER állapot, X'40' - nem volt még aktiv gomb. Az alsó hat bit a fizikai cím (ld. A. melléklet).

+3. (1) Klaviatúra/display flag

Bit 0 - DROW rutin - inverz vonalhúzás

Bit 1 - DROW rutin - ha bit0=0, akkor a vonalat  
1=rajzolja, 0=törli

Bit 2 - 20 perces TIMER aktiv

Bit 3 - Funkció billentyű scan (speciális üzemmód-  
ban a 'lenyil' gomb le van nyomva)

Bit 4 - Printer X'1E', X'1F' konverzió tiltás. A  
két karakter a hosszú kis i, és a felfele  
nyil.

Bit 5 - az ernyőn levő cursor éppen világít

Bit 6 - Break tiltás - ha 1, akkor BRK(=1) helyett  
0 kód jön be.

Bit 7 - Klaviatúra speciális üzemmód. Hatására a  
'lenyil' shift gombként dolgozik, és a kódok  
értékéhez X'40'-et ad hozzá. A bitet az  
X'0035' - Linein funkció használja.

+4. (1) Klaviatúra időzítés. (default=24, lehet 10-255)

+5. (1) Display flag

Bit 0 - Duplaszéles karakter

Bit 1 - Aláhúzás

Bit 2 - Inverz ernyő

Bit 3 - Inverz karakter

Bit 4 - Pre-clear

Bit 5 - Pre-clear puffer

Bit 6 - Inverz rajzolás



Bit 7 - Függőleges írás

- +6. (1) A display kurrens X koordinátája. Ha V a karakterben mért koordináta, akkor  $X=4+V*6$ , ahol  $V=0-41$ . A rekesz kezdőértéke 4. (Mivel X irányban 0-255 pont van, és 256 nem osztható a karakterek 6 pont szélességével, ez egyenliti ki a differenciát.)
- +7. (1) A display kurrens Y koordinátája. Ha W a sorban mért koordináta, akkor  $Y=1+W*12$ , ahol  $W=0-15$ . A rekesz kezdőértéke 1, mivel a karakterek első sora mindig üres (scrkőz). Az Y értéke 0-191 lehet. A hardware címzés a bal felső sarokból (0,0) indul.
- +8. (1) Az ernyőpuffer (6 kbyte) kezdőcíme. Mivel az alsó 11 bit értéke mindig 0, itt a címnek csak a felső byte-ja tárolódik. Az ernyőpuffer címe csak két érték lehet: a memória utolsó 6 kbyte-ja, vagy egy ennél 8 kbyte-al kisebb érték. (Ennek hardware okai vannak, ugyanis a hardware ernyőpuffer átkapcsolás csak a cím egy bitjének állítására képes, a többi huzalozott.)
- +9. (2) A 128 (X'80') feletti kódú karakterek generátortáblájának címe.
- +11. (1) Pattern regiszter
- +12. (1) Pattern puffer regiszter
- +13. (1) Üres
- +14. (1) A nyomtató maximális sorhossza.
- +15. (1) A nyomtató maximális laphossza. Ha az itt megadott érték nagyobb, mint 127, nincs laphossz ellenőrzés.
- +16. (1) A nyomtató soron belüli maradékszámológója.
- +17. (1) A nyomtató lapon belüli maradékszámológója.

- +18. (1) A nyomtató soron belüli pozíciószámlálója.
- +19. (1) Üres
- +20. (1) IOSTAT byte a kazettakezelés részére.
  - Bit 0 - Nyitott kazettafile (1)
  - Bit 1 - Input(1) vagy output (0)
  - Bit 2 - A puffer üres (0), vagy kitöltve (1)
  - Bit 3 - TEST (1), vagy LOAD (0) funkció
  - Bit 4 - Hangadás ( $\phi$ ), vagy nincs ( $\Lambda$ )
  - Bit 5 - Filenév van a kereséshez (1), vagy nincs
  - Bit 6 - Szövegírás az ernyőre (0), vagy nincs (1)
  - Bit 7 - Üres
- +21. (1) IOCNTNTR byteszámláló, 0-tól 0-ig számol
- +22. (2) I/O puffer pointer, az első szabad byte-ra mutat.
- +24. (2) I/O puffer kezdőcím
- +26. (1) Magnó rekordtípus.
- +27. (1) Magnó hibás rekord számláló. értéke 0-ról indul, és BCD-ben növekszik. (azaz 0-99 lehet)
- +28. (1) Magnó rekordszámláló. értéke 0-ról indul, és BCD-ben növekszik. (azaz 0-99 lehet)
- +29. (1) Magnó fázisfordítás jelző. Ha itt X'10' van, a magnó nem fordít fázist, ha X'EF', akkor igen.
- +30. (1) Magnó időalapja.
- +31. (1) Üres

#### 4.2 Egyéb SYSRAM címek

(000B) - *sysram end + 1*  
(0013) - *screen start*  
(001B) - *DCR addr*

(X'0023) - X'403D' (jelenleg) - REAL-TIME óra. Ez a három byte-os egész szám (legelső byte a legalacsonyabb helyiérték, a legutolsó a legmagasabb) bekapcsoláskor 0-ról indul, és minden 0.02 másodpercben (pontosabban  $312 \cdot 64 \text{ usec} = 19.968 \text{ msec}$ ) 1-el növekszik. Az NMI kikapcsolása leállítja.

X'007E' - Generálási dátum, jelenleg X'84'

X'007F' - Verziószám, jelenleg X'01'

*Y'0180'* - *hazakutasi verziószám*

X'403E' - OUTBYTE. A hardware vezérlésére szolgál. Kiküldése a 0-63 címek közül bármelyikre történhet. Mivel a hardware-ből visszaolvasni nem lehet, ezért itt van a parallel nyilvántartás. Kiosztását ld. az Input/Output leírásban. Induláskor értéke 0-ra áll be.

Az RST utasítások valamennyien a RAM-ba ugranak ki, funkciójuk elvégzése előtt. A megadott címeken egy-egy JP utasítás található, amely tovább ugrik a funkciórutinra. Az egyes RST utasítások címe és funkciója:

RST 00H - Bekapcsolási törlés. Az X'000D' -t hívja.

RST 08H - X'4000' - BASIC

RST 10H - X'4003' - BASIC

RST 18H - X'4006' - BASIC

RST 20H - X'4009' - BASIC

RST 28H - X'400C' - Szabadon felhasználható

RST 30H - X'400F' - Szabadon felhasználható



RST 38H - X'4012' - Fenntartva az IM1-es IT-knek. Jelenleg itt jelenik meg a COMMODORE BUS Serv. Req jele, és a magnó input a 0-ba váltás esetén. (A COMMODORE-hoz be kell kötni egy opcionális diódát)

Egyéb ugrási címek:

RESET - X'401E' - A RESET gomb a SYSRAM és a STACK initje után az itt található JP utasításra ugrik. Így a program le tudja kezelni a RESET gombot, de a megszakítás helyére visszatérni nem lehet a STACK initje miatt. A RESET-et hatálytalanítja az NMI letiltása (ld. OUTBYTE), de ez a REAL-TIME órát is kikapcsolja.

SSPEC - X'4027' - Speciális klaviatúra üzemmódban, ha a 'lenyil' + karakter sorozatot nyomjuk, kód+X'40' generálódik, majd ugrás történik az adott címre. Itt egy JP utasítás van egy return-re. Így lehetőség van PF (Prog.Func) gombok definiálására.

A jelenleg fel nem használt RST utasítások RAM rekeszeiben JP 3148H található (ugrás egy RET utasításra), míg a többiben a megfelelő rutinra ugrás szerepel. (később ez JP 148H-ra változik)

NMI vége X'4018' HL is A a stacken

POP HL  
POP AF  
RET

} 4. lépés,  
ha RESET, oda ugrik

#### 4.3 BASRAM címek

- X'40A2'* - *első BASIC sorozat (ha parancsok, kezdés &FFFF)*
- X'40AC'* - STRING memória eleje. A STACK kezdőcíme =  $(X'40A0')-2$
- X'40A4'* - BASIC programok kezdőcíme. Ha átírjuk, a tár elejéből kapunk más célra (pl. assembler programok) felhasználható területet. Átirása esetén ki kell adni a NEW parancsot, azaz a bentlevő programot törölni kell.
- X'40AA'-X'40AC'* - BASIC END függvény kezdőértéke. A kezdőérték 3 byte-os egész formájában tárolódik LSB,M,MSB sorrendben.
- X'40E1'* - BASIC STRING memória végcíme. Ha átírjuk, a tár végéből kapunk más célra (pl. assembler programok) felhasználható területet. Átirása esetén ki kell adni a CLEAR parancsot, amely gondoskodik a BASIC STACK áthelyezéséről, és a STRING memória újra allokálásáról. (Vigyázat ! Csak akkor, ha a CLEAR-t paraméterrel adjuk ki !)
- X'40D6'* - STRING kurrens pointer.
- X'40E8'* - A BASIC minden funkciórutin hívás előtt ide menti az SP-t.
- X'40F9'* - BASIC skaláris változók kezdőcíme. A BASIC program mentéséhez ennek tartalma lesz a végcím a kazettakezelő számára.

X'40FB' - BASIC tömbváltozók kezdőcíme. Vigyázat ! ez a cím a programfutás során dinamikusan változik !

X'40FD' - BASIC tömbváltozók végcíme+1. Ez adja meg a BASIC program utáni első szabad címet. Vigyázat ! ez a cím a programfutás során dinamikusan változik !

X'40FF' - DATA pointer (ind: = (40A4) azaz előző)



## 5. INPUT/OUTPUT

Az I/O lényegében két input és két output port-ból áll, továbbá ide számítható a képernyő kezelés is.

Az output címek kiosztása:

B'00xxxxxx' (0-63) - kezdőérték 0 (ld. még SYSRAM X'403B').

Kiosztása:

Bit 0 - NMI engedélyezés (1)/tiltás(0). Letiltott NMI esetén nincs REAL-TIME óra, és nem működik a RESET gomb.

Bit 1 - Joystick számláló léptetés (0-4 hasznos lekérdezés, 5-7 üres léptetés - ld. input 64-127)

Bit 2 - Magnó távvezérlés a tuchel 4. pontra, V24-nek is ( $\pm$  5V) köthető.

Bit 3 - Hangszóró impulzus(1) ill. szünet (0)

Bit 4 - Display puffer vezérlés. (1) - magasabb cím (normal), (0) - alacsonyabb cím

Bit 5 - Magnó távvezérlés a tuchel 5. pontra, V24-nek is ( $\pm$  5V) köthető. A bit a BUS A-22 pontján is megjelenik.

Bit 6-7 - Magnó illesztő vezérlés a tuchel 1. pontján. A két bit analóg összegzéssel jelenik meg, azaz 00 - 0 szint, 01/10 1 (közép) szint, az 11 pedig 2 (magas) szint. Így lehet négyszög hullámot programozni.

B'01xxxxxx' (64-127) - kezdőérték 0. Kiosztása:

Bit 0 - üres

Bit 1 - üres

Bit 2 - COMMODORE Serial Clock

Bit 3 - COMMODORE Serial Data

Bit 4 - üres

Bit 5 - üres

Bit 6 - COMMODORE ATTN

Bit 7 - üres

Az input címek kiosztása:

B'00xxxxxx' (0-63) A cím x-el jelölt része a klaviatúra 59 billentyűjét címzi. (a címeket ld. az A. mellékletben) A kapott input byte kiosztása:

Bit 0-1 - üres

Bit 2-3 - bővítésre fenntartva

Bit 4 - (1), ha az ernyő végén vagyunk

Bit 5 - magnó input a tuchel 3. pontjáról, ill. V24 input. A 0-ba átmenet megszakítást (RST 38H) okoz.

Bit 6 - Reset gomb be van nyomva (1) vagy nem (0)

Bit 7 - A címzett klaviatúra gomb le van nyomva (1) vagy nem (0)

B'01xxxxxx' (64-127) A cím x-el jelölt része tetszőleges. a kapott input byte kiosztása:

Bit 0 - üres

Bit 1 - üres

Bit 2 - COMMODORE Serial Clock

Bit 3 - COMMODORE Serial Data

Bit 4 - COMMODORE Serv Req (RST 38H IT-t okoz, ha az opcionális dióda be van kötve)

Bit 5 - Joystick 2. A kurrens léptetéstől függően (ld. output 0-63) beolvassa a joystick mikrokapcsolóinak állását. Az öt léptetési állapotban a négy irány és a lövést jelző kapcsoló állását kapjuk. A megfelelő olvasáshoz a számlálót végig kell léptetni mind a nyolc állapoton.

Bit 6 - COMMODORE ATTN

Bit 7 - Joystick 1. (ld. bit 5.)

#### Ernyőkezelés:

Az ernyőkezeléshez a PRIMO a memória utolsó 16 kbyte-ját speciálisan figyeli. A processzor és az ernyőfrissítés összeütközéseit mindig az ernyő javára oldja fel. Az ernyő mérete a hardware-ben jumperrel állítható, vagy 6 kbyte (normál  $192 \times 256 \Rightarrow 42 \times 16$ ) vagy 8 kbyte ( $256 \times 256 \Rightarrow 42 \times 21$ ). Az utóbbi bekötést a software nem támogatja, így az ernyő tetején sötét csík marad. Átkötés esetén az ernyő címe  $X'E000'$  (ill.  $X'A000'/X'6000'$ ), és gondoskodni kell a BASIC végcím megfelelő átirásáról. Az alternatív puffercím is változik  $X'C000'/X'8000'$  -ra (Az A16 változatnál a képernyő nem váltható).



6. KARAKTERGENERÁTOR VEZÉRLÉS

A 128 (X'80') feletti karakterek képét meg lehet változtatni. Ehhez saját táblát kell összeállítani, és címét a DCB+9, DCB+10 mezőbe kell helyezni (alacsony helyiérték elől). Az egy karakter által elfoglalt mező képe 6\*12-es:

Normál	Sülylesztett
+-----+	+-----+
*	
*****	
*	
*	*    *
***	*    *
*	*    *
*	****
*****	*
	*    *
	***
+-----+	+-----+

A karakterdefiníció általában 6\*8-os méretű. A legfelső sort nem lehet átírni, azt a karaktergenerátor mindig üresen hagyja. A 8 sorból az első az ékezetre, a hat oszlopból az utolsót pedig a betűközre szokás fenntartani. Az un. sülylesztett karaktereket külön meg kell jelölni, és azokat a karaktergenerátor 2 pozícióval lejjebb sülylesztí (ld. az ábrát). A karakter képét általában 8 byte írja le, ezek felépítése:

- Byte 0 bit 0 Ha értéke 1, akkor a karakter sülylesztett.
- bit 1-6 A karakter első sora. Ahol 1 érték van, ott lesz fénypont.
- bit 7 Ha egy, a karakter végét jelzi.
- Byte n bit 0 értéke közömbös.
- bit 1-6 A karakter n.-dik sora. Ahol 1 érték van,

ott lesz fénypont.

bit 7      Ha egy, a karakter végét jelzi.

Megjegyezzük, hogy ha egy karakterleírás hosszabb, mint 8 byte, többsoros karaktert definiálunk, amely akár több karaktersoron ( $n \cdot 12$  bitsoron) át is húzódhat. Ennek azonban több karakterkód elfoglalása az ára, mert ha a leírás  $n \cdot 8$  hosszú, a kezdőkódot követő  $n-1$  kód nem használható.

## 7. KAZETTAFORMATUM

A kazetta file három részből áll: header, adat(ok), és trailer. A file lehet szabványos, ekkor a típusokat szigorúan be kell tartani, és load paranccsal, illetve BASIC-ből a file olvasható lesz. A nem szabványos típusok csak speciális assembler programmal olvashatóak. Két alaptípus a program és az adatfile. Az adatfile nem keveredhet semmi mással, a programfile azonban vegyesen tartalmazhat BASIC és Assembler programot, valamint elmentett ernyőképet (SCREEN). Az egyes blokkok felépítése:

Header    512 db. X'AA' - file szinkron

          96 db. X'FF' + 3 db. X'D3' - rekord szinkron

          1 byte rekordtípus

                  - X'83' - programfile

                  - X'87' - adatfile

          1 byte rekordsorszám - értéke 0

          1 byte névhossz - n=1-16

          n byte név

          1 byte kontrolösszeg - a rekordsorszámtól kezdve,

                  és azt beleértve 8 bites összeg

Data        96 db. X'FF' + 3 db. X'D3' - rekord szinkron

          1 byte rekordtípus

                  - X'F1' - BASIC program

                  - X'F5' - SCREEN

                  - X'F9' - Assembler

                  - X'F7' - adatfile



1 byte rekordsorszám (modulo 100) BCD-ben

2 byte töltési cím (alacsony/magas sorrendben) -  
BASIC programok és SCREEN esetén mindig  
relatív cím szerepel, amely 0-tól indul. Az  
adatfile-ok minden blokkja 0-ra van címezve.  
Az assembler programok a valós betöltési  
címet tartalmazzák.

1 byte hossz (0 megadása 256-ot jelent)

n byte adat (1-256) - Az adatfile-okban az elemek  
vesszővel vagy X'0D' (CR) karakterekkel van-  
nak elválasztva a PRINT# utasításnak megfe-  
lelően. Az egyes elemek átcSORoghatnak a  
blokkhatárokon.

1 byte kontrolösszeg - a rekordsorszámtól kezdve,  
és azt beleértve 8 bites összeg

Trailer 96 db. X'FF' + 3 db. X'D3' - rekord szinkron

1 byte rekordtípus

- X'B1' - BASIC vagy Assembler program
- X'B5' - SCREEN,
- X'B9' - autostartos Assembler program
- X'B7' - adatfile

1 byte rekordsorszám (modulo 100) BCD-ben

2 byte indítási cím (alacsony/magas sorrendben) -  
csak X'B9' típus esetén.

1 byte kontrolösszeg - a rekordsorszámtól kezdve,  
és azt beleértve 8 bites összeg

## 8. ASSEMBLERBŐL HÍVHATÓ RUTINOK

### 8.1 Display kezelés

```
+-----+
|         |
| DISPLAY |
|         |
+-----+
```

Display X'0015' Egy karakter írása a képernyő kurrens pozíciójába. Utána a kurrens pozíció módosul (ld. DCB+6,7). A rutin az ernyővezérlő kódokat értelmezi. Minden regisztert ment.

Input: A= kiírandó karakter

Output: A= változatlan, F= beáll - (OR A).

```
+-----+
|         |
| INKEYE  |
|         |
+-----+
```

INKEYE X'001D' Egy karakter olvasása a klaviatúráról várakozás nélkül. A kurrens pozíció nem módosul, a karakter nem íródik vissza. Valamennyi kód bejön, beleértve a BRK-t is. Ha a klaviatúrán nincs semmi lenyomva, 0-t ad. Minden regisztert ment.

Input: --

Output: A= karakter vagy 0, F= beáll - (OR A).

```

+-----+
|           |
|   INPUT   |
|           |
+-----+

```

**INPUT X'0025'** Egy karakter olvasása a klaviatúráról vára-  
kozással. A kurrens pozíció nem módosul, a  
karakter nem íródik vissza. Valamennyi kód  
bejön, beleértve a BRK-t is. A kurrens  
pozícióba kiírja a CURSOR-t, és vár. Elegen-  
dő sűrűséggel hívva a REPEAT funkció is  
működik. Minden regisztert ment.

Input: --

Output: A= karakter, F= beáll - (OR A).

```

+-----+
|           |
|  LINEIN   |
|           |
+-----+

```

**Linein X'0035'** Egy sor olvasása a klaviatúráról visszairás-  
sal és sorszerkesztéssel. A funkció végét a  
max. 210 karakter beérkezése, a RETURN,  
illetve a BRK okozzák. A sor végét a puffer-  
ben egy nulla byte jelzi. BRK megnyomása  
esetén mindig üres sort kapunk. A vezérlőkó-  
dok értelmeződnek, de nem kerülnek a puffer-  
be. A 'lefele nyíl' speciális SHIFT-ként  
dolgozik, a vele együtt lenyomott billentyűk  
a szokásos kódjuk helyett ennél X'40'-nel  
nagyobb kódot adnak be. A már beírt sort  
törölni lehet a CLS gombbal, ill. a SHIFT  
<-- gombbal. A balra nyíl egy karakter tör-



lését okozza. A kurrens pozícióba kiírja a CURSOR-t, és vár a következő karakterre. A kurrens pozíció a beírásnak megfelelően módosul. A REPEAT funkció működik. Minden regisztert ment.

Input: HL= puffercim - 1

Output: HL= változatlan, A= X'0D', F= CY, ha  
BRK volt.

```
+-----+
|       |
|  DCURS  |
|       |
+-----+
```

DCURS X'0080' Cursor címzés rutin. Az átadott paramétereknek megfelelően beállítja a kurrens pozíciót a DCB+6,7-ben.

Input: D= 0-15 sorcim, E= 0-41 pozíció

Output: Az A, F -et rontja

```
+-----+
|       |
|   SET   |
|       |
+-----+
```

SET X'0083' A megcímzett ernyőpontot fehér alapon eloltja, sötét alapon kigyújtja. A címzésnél a bal alsó sarok a (0,0). Ha az y>191, akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál,  
A, C romlik.

```

+-----+
|       |
|  RESET  |
|       |
+-----+

```

RESET X'0086' A megcímzett ernyőpontot sötét alapon eloltja, fehér alapon kigyújtja. A címzésnél a bal alsó sarok a (0,0). Ha az  $y > 191$ , akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál, A, C romlik.

```

+-----+
|       |
|  POINT  |
|       |
+-----+

```

PCINT X'0089' A megcímzett ernyőpontot megvizsgálja. A fehér alapon sötét ill. sötét alapon fehér pont esetén jelez 'igaz' feltételt. Igaz feltétel esetén nem zérus, egyébként zérus feltétel áll be az F regiszterben. A címzésnél a bal alsó sarok a (0,0). Ha az  $y > 191$ , akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál, Z a hamis feltételnél, A, C romlik.

```

+-----+
|       |
|  POS  |
|       |
+-----+

```

POS      X'008C'    Visszaadja a kurrens soron belüli karakter-  
pozíciót.

Input:    --

Output:   A= pozíció (0-41)

## 8.2 Printer kezelés

```

+-----+
|       |
| PRINT |
|       |
+-----+

```

Print    X'002D'    Egy karakter írása a printer kurrens pozíci-  
ójába. Utána a kurrens pozíció módosul (ld.  
DCB+16,17,18).    A kimenő vezérlőkódok közül  
az X'0D', X'0A'    - soremelés, és az X'0C' -  
lapdobás értelmeződik. A többi vezérlőkód  
változtatás nélkül kiíródik. Az X'1E', X'1F'  
kódok (hosszú kis i, felfele nyíl) a DCB+3  
szerint konvertálódhatnak. Minden regisztert  
ment. A nyomtatón a helyes működés érdeké-  
ben a svéd karakterkészletet kell beállíta-  
ni.

Input:    A= kiírandó karakter

Output:   A= változatlan,    F= beáll    A szerint  
(OR A) .



### 8.3 Ernyőpuffer kezelés

```
+-----+
|       |
|  ALLOC  |
|       |
+-----+
```

**Alloc**    **X'3123'** Második ernyőpuffert allokál a buff-8kbyte címre. Az allokálást csak 16kbyte RAM-nál nagyobb gépben végzi el. Ezt a területét a BASIC-től elveszi, így kiadása után CLEAR parancs szükséges. A két ernyőpuffer közötti 2kbyte mezőt tetszőleges célra fel lehet használni.

**Input:**    --

**Output:** A,F elromlik.

```
+-----+
|       |
| SWITCH |
|       |
+-----+
```

**Switch**   **X'312D'** Csak már allokált második ernyőpuffer esetén működik. Kivárja az ernyőfrissítés végét, majd mind a hardware-ben, mind a DCB+8 -ban átírja az ernyőpuffer címét. A funkció oda-vissza kapcsol a két ernyőpuffer között.

**Input:**    --

**Output:** A,F elromlik.

## 8.4 Kazetta kezelő rutinok



**Wrhdr** X'0092' File header írására szolgál. A DCB+26 -ba hívás előtt be kell írni a header típust. Komplet headert ír ki. A HL -ben kell átadni a file név leíró címét. Ennek felépítése megegyezzeik a standard string leírással (első byte a hossz, második és harmadik byte a string címe, alacsony/magas helyiérték sorrendben). Elindítja a magnót, felvesz 4 sec szünetet, majd felírja a header-t.

Input: HL= névleírás, DCB+20= hangadás,  
DCB+26= típus

Output: HL, DCB+26 változatlan, a DCB+28= 0  
(rekordszám) AF, BC -t ront.



**Wrdata** X'0095' File adat írására szolgál. A DCB+26 -ba hívás előtt be kell írni az adattípust. Komplet adatot ír ki, a DCB+28 (rekordsorszám) automatikusan módosul. Egyszerre akár-hány blokk kiírható vele. A file-ba töltési címként a memóriacím-offset érték kerül. BASIC program kimentésénél offset = kezdő-

cím.

Input: DE= kezdőcím, HL= végcím+1, BC= offset,  
DCB+20= hangadás, DCB+26=  
tipus, DCB+28= rekordsorszám (BCD)

Output: DCB+28 inkrementálódik a kiírt blokkoknak megfelelően. AF, DE, HL romlik

```
+-----+  
|      |  
|  WRTL  |  
|      |  
+-----+
```

Wrtl X'009B' File trailer írására szolgál. A DCB+26 -ba hívás előtt be kell írni a header típust. Komplet trailert ír ki, majd leállítja a motort. Automatikus indítású programhoz nem használható, mert az indítási címet nem tudja felírni. Ilyen esetben byte-onkénti kivittelt kell használni. A rekordsorszámot növeli.

Input: DCB+20= hangadás, DCB+26= típus,  
DCB+28= rekordsorszám (BCD)

Output: DCB+28 megnövelve. AF, BC romlik

```
+-----+  
|      |  
|  WRSYN  |  
|      |  
+-----+
```

Wrsyn X'3ADD' File szinkron írására szolgál. Kiír 96 db. X'FF'-et, és 3 db. X'D3'-at.

Input: --



Output: --, AF, BC romlik

```
+-----+
|       |
| WRBYTE |
|       |
+-----+
```

Wrbyte X'3AF4' Egy karaktert ír a magnóra, kontrolösszeg számítás nélkül. A DCB+20 X'08' bitje (1/0) engedélyezi/tiltja a hangadást.

Input: A= kiírandó karakter, DCB+20= hangadás vezérlés

Output: --, AF romlik

```
+-----+
|       |
| WRBYTC |
|       |
+-----+
```

Wrbytc X'3AF0' Egy karaktert ír a magnóra, kontrolösszeg képzéssel. A DCB+20 X'08' bitje (1/0) engedélyezi/tiltja a hangadást. A kiírt karakter hozzáadódik a C regiszter tartalmához. Az első hívás előtt C-t nullázni kell, majd a további hívásokhoz meg kell őrizni. A blokk végére ki kell írni a Wrbyte funkcióval a C-ben levő kontrolösszeget. A rekordszám az első olyan byte, amit bele kell számolni a kontrolösszegbe.

Input: A= kiírandó karakter, C= előző kontrolösszeg, DCB+20= hangadás vezérlés

Output: C= módosított kontrolösszeg. AF romlik

```

+-----+
|       |
|  DRVON  |
|       |
+-----+

```

Drvon X'3B24' A magnómotor bekapcsolására szolgál.

Input: --

Output: AF-t rontja

```

+-----+
|       |
|  DRVOFF  |
|       |
+-----+

```

Drvcff X'3B26' A magnómotor kikapcsolására szolgál.

Input: --

Output: AF-t rontja

```

+-----+
|       |
|  SRCHDR  |
|       |
+-----+

```

Srchdr X'008F' File header keresésére szolgál. A magnót indítja, majd 20 msec csendet vár. Ezután keresi a file szinkront, időalapot mér, és megállapítja, hogy szükséges-e fázisfordítás. Ezután keresi a rekordszinkront, beolvassa a headert, és ha volt név megadva, teszteli. Ha kell, üzenetet ír a displayra, ahol beállítja a pre-clear üzemmódot.

Input: DCB+20, DCB+26, HL= filenév leíró címe.

Output: DCB+27,29,30 beállítva

```

+-----+
|       |
| SRC SYN |
|       |
+-----+

```

Srcsyn X'3C75' Rekord szinkron keresésére szolgál. Ha talált, beolvassa a rekordtipust.

Input: --

Output: A= rekordtípus

```

+-----+
|       |
| IN BYTE |
|       |
+-----+

```

Inbyte X'009E' Beolvas egy byte-ot, és növeli a kontrolösszeget.

Input: D= eddigi kontrolösszeg

Output: A= input byte, D= új kontrolösszeg

```

+-----+
|       |
| CHK SUM |
|       |
+-----+

```

Chksum X'3C22' Beolvassa a kontrolösszeget, és összehasonlitja a D-vel. Ha kell, növeli a hibaszámlálót, és kiírja a rekordszámmal együtt a display-re.

Input: D= eddigi kontrolösszeg, E= rekord-sorszám, DCB+27

Output: DCB+27



```

+-----+
|          |
|  RDDATA  |
|          |
+-----+

```

Rddata X'0098' File adat olvasására szolgál. Komplet  
blokkot olvas be. Előtte hívni kell a  
rekordszinkron keresést, majd be kell olvas-  
ni a rekordsorszámot. (ehhez D= 0 kell, mert  
a sorszám már beleszámít a kontrolösszegbe.)  
A DCB+27 automatikusan módosul. A blokk a  
memóriába a töltési cím + offset címre  
kerül. A checksum-ot ellenőrzi.

Input: D= előző kontrolösszeg ( rekordsor-  
szám ) E= rekordsorszám (BCD), BC=  
offset, DCB+27

Output: DCB+27,28, HL=HL+BC, BC, DE romlik

## 8.5 Egyéb rutinok

```

+-----+
|          |
| PWRESET  |
|          |
+-----+

```

PWRESET X'000D' POWER-ON reset rutin, nem tér vissza. Ini-  
cializálja a SYSRAM-ot, majd a BASIC init-et  
hívja.

Input: --

Output: -- nem tér vissza !

```

+-----+
|       |
|  RELOC  |
|       |
+-----+

```

Reloc    X'002B' Később kerül implementálásra. Önáthelyező programok támogatására szolgál. Hívásakor megadja a hívó CALL utasítást követő címet.

Input:    --

Output: HL= a hívó CALL utáni cím.

```

+-----+
|       |
|  CLRS   |
|       |
+-----+

```

CLRS    X'1B61' A BASIC CLEAR funkcióját hajtja végre a STACK átállítással együtt. Hívásához be kell állítani a STRING memória elejének, és végének címét a BASRAM-ban. (X'40A0', X'40B1', X'40D6')

Input:    BASRAM rekeszek.

Output: BC-t rontja

```

+-----+
|       |
|  NEWC   |
|       |
+-----+

```

NEWC    X'1B4A' A BASIC NEW funkcióját hajtja végre, ernyő-törléssel együtt.

Input:    --

Output:    --

NEW
-----

NEW X'1B4D' A BASIC NEW funkcióját hajtja végre, ernyő-törlés nélkül.

Input: --

Output: --

CLEAR
-------

CLEAR X'1E83' A BASIC CLEAR funkcióját hajtja végre a STACK átállítással együtt.

Input: DE= stringterület új hossza

Output: BC-t rontja

VARPTR
--------

VARPTR X'2664' BASIC skaláris változók címének megkeresésére szolgál. Alapos tesztelést kíván, használata bizonytalan. Még nem allokált változó esetén az allokálást is elvégzi.

Input: BC= név, D= típus (2= INT, 3= STR, 4= SNG, 8= DBL)

Output: DE= a változó címe



```

+-----+
|       |
|  BEEP  |
|       |
+-----+

```

BEEP     X'3F68' A BASIC BEEP rutint hívja. Leírását ld. ott.

BC= időtartam, DE= T/2

Output: AF, BC romlik

```

+-----+
|       |
|  RUN   |
|       |
+-----+

```

RUN     X'1B5D' A BASIC programot indítja. Nem tér vissza a hívóhoz. A rutint nem CALL, hanem JP utasítással hívjuk, és hívás előtt a STACK-re kell tenni X'1D1E'-t.

Input:    STACK= X'1D1E'

Output: -- , nem tér vissza.

```

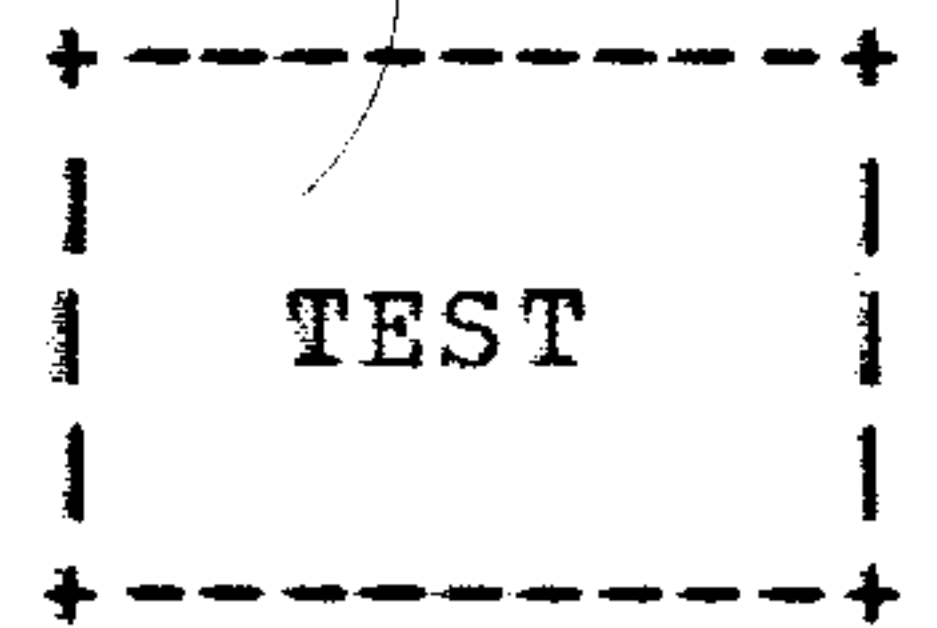
+-----+
|       |
|  LOAD  |
|       |
+-----+

```

LOAD     X'050F' A BASIC LOAD rutinját hívja. Ha a program automatikusan indul, nem tér vissza a hívóhoz. Híváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva. Ha a nevet nem adjuk meg, egy üres soremelést kell átadni.

Input:    (HL) -> "név", X'0D'

Output: ??? romlik



TEST X'059B' A BASIC TEST rutinját hívja. Híváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva. Ha a nevet nem adjuk meg, egy üres soremelést kell átadni.

Input: (HL)-> "név", X'0D'

Output: ??? nem tér vissza



SAVE X'04CD' A BASIC SAVE rutinját hívja. Híváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva.

Input: (HL)-> "név", X'0D'

Output: ??? nem tér vissza

## Klaviatúra címek kód szerint rendezve

0	0	Y	23	17	7 /	43	2B	. :
1	1	A	24	18	H	44	2C	M
2	2	S	25	19	SPACE	45	2D	9 )
3	3	SHIFT	26	1A	E	46	2E	I
4	4	E	27	1B	6 &	47	2F	, ;
5	5	UPPER	28	1C	G	48	30	Ü
6	6	W	29	1D	5 %	49	31	' *
7	7	CTR	30	1E	V	50	32	P
8	8	D	31	1F	4 E	51	33	ű ú
9	9	3 #	32	20	N	52	34	O
10	A	X	33	21	8 (	53	35	CLS
11	B	2 "	34	22	Z	55	37	RETURN
12	C	Q	35	23	+ ?	57	39	<--
13	D	1 !	36	24	U	58	3A	é
14	E	A	37	25	0 =	59	3B	ó ô
15	F	V	38	26	J	60	3C	Á
16	10	C	39	27	> <	61	3D	-->
18	12	F	40	28	L	62	3E	ő
20	14	R	41	29	- i	63	3F	BRK
22	16	T	42	2A	K			

## Klaviatúra címek karakter szerint rendezve

. :	43	2B	C	16	10	T	22	16
<--	57	39	D	8	8	U	36	24
+ ?	35	23	E	4	4	Ü	48	30
A	1	1	é	58	3A	ű ú	51	33
V	15	F	F	18	12	V	30	1E
- i	41	29	G	28	1C	W	6	6
-->	61	3D	H	24	18	X	10	A
, ;	47	2F	I	46	2E	Y	0	0
> <	39	27	J	38	26	Z	34	22
' *	49	31	K	42	2A	0 =	37	25
BRK	63	3F	L	40	28	1 !	13	D
CLS	53	35	M	44	2C	2 "	11	B
CTR	7	7	N	32	20	3 #	9	9
RETURN	55	37	O	52	34	4 E	31	1F
SHIFT	3	3	ó ô	59	3B	5 %	29	1D
SPACE	25	19	ő	62	3E	6 &	27	1B
UPPER	5	5	P	50	32	7 /	23	17
A	14	E	Q	12	C	8 (	33	21
A	60	3C	R	20	14	9 )	45	2D
B	26	1A	S	2	2			



## Karakterkészlet

0	0		51	33	3	102	66	f
1	1	CTL-A, BRK	52	34	4	103	67	g
2	2	CTL-B	53	35	5	104	68	h
3	3	CTL-C	54	36	6	105	69	i
4	4	CTL-D	55	37	7	106	6A	j
5	5	CTL-E	56	38	8	107	6B	k
6	6	CTL-F	57	39	9	108	6C	l
7	7	CTL-G, BELL	58	3A	:	109	6D	m
8	8	CTL-H, <--	59	3B	;	110	6E	n
9	9	CTL-I, SH -->	60	3C	<	111	6F	o
10	A	CTL-J, V	61	3D	=	112	70	p
11	E	CTL-K	62	3E	>	113	71	q
12	C	CTL-L, CLS	63	3F	?	114	72	r
13	L	CTL-M, RETURN	64	40	é (U)	115	73	s
14	E	CTL-N, CR	65	41	A (U)	116	74	t
15	F	CTL-O	66	42	B (U)	117	75	u
16	10	CTL-P	67	43	C (U)	118	76	v
17	11	CTL-Q	68	44	D (U)	119	77	w
18	12	CTL-R	69	45	E (U)	120	78	x
19	13	CTL-S	70	46	F (U)	121	79	y
20	14	CTL-T	71	47	G (U)	122	7A	z
21	15	CTL-U	72	48	H (U)	123	7B	ő
22	16	CTL-V	73	49	I (U)	124	7C	ö
23	17	CTL-W	74	4A	J (U)	125	7D	á
24	18	CTL-X, SH <--	75	4B	K (U)	126	7E	ü
25	19	CTL-Y, -->	76	4C	L (U)	127	7F	ű
26	1A	CTL-Z, SH V	77	4D	M (U)	128	80	Pi
27	1E		78	4E	N (U)	129	81	sz. (
28	1C	CTL-ö	79	4F	O (U)	130	82	sz.)
29	1I	CTL-Á	80	50	P (U)	131	83	<- (K)
30	1E	CTL-Ü, i	81	51	Q (U)	132	84	-> (K)
31	1F	▲	82	52	R (U)	133	85	V (K)
32	20	SPACE	83	53	S (U)	134	86	Gyök
33	21	!	84	54	T (U)	135	87	Pipa
34	22	"	85	55	U (U)	136	88	Integrál
35	23	#	86	56	V (U)	137	89	Ft
36	24	E	87	57	W (U)	138	8A	
37	25	%	88	58	X (U)	139	8B	ü
38	26	&	89	59	Y (U)	140	8C	Paragraph
39	27	'	90	5A	Z (U)	141	8D	†
40	28	{	91	5B	ó	142	8E	Á
41	29	}	92	5C	ö (U)	143	8F	
42	2A	*	93	5D	Á (U)	144	90	
43	2E	+	94	5E	Ü (U)	145	91	Hullám
44	2C	,	95	5F	ú	146	92	Omega
45	2D	-	96	60	é	147	93	Mű
46	2F	.	97	61	a	148	94	Négyzet
47	2F	/	98	62	b	149	95	Azonos
48	30	0	99	63	c	150	96	Szumma
49	31	1	100	64	d	151	97	Produktjel
50	32	2	101	65	e			

A (K) a nyilak nyomtatási kódját jelzi. Az (U) a nagybetűk mellé került a táblázatban.

## Ernyővezérlő kódok

0	0		A BASIC string végjelnek használja, kiírva nincs pozíciótovábbítás
1	1	CTL-A, BRK	BASIC programfutás megállítása, kiírva alaphelyzetbe állítja a képernyőkezelő opciókat.
2	2	CTL-B	Duplaszéles karakterek írása (21 char/scr)
3	3	CTL-C	Az ernyő alapszíne fehér lesz, és a karakterek feketén jelennek meg.
4	4	CTL-D	A karakter háttérszíne az ernyő ellentettje lesz, a karakter ehhez képest inverzen jelenik meg.
5	5	CTL-E	A karakterek aláhúzva jelennek meg.
6	6	CTL-F	A karakterek helye kiírás előtt törlődik
7	7	CTL-G, BELL	Megszólaltatja a dudát
8	8	CTL-H, <--	Kurzens pozíció törlése visszalépéssel
9	9	CTL-I, SH -->	Tabulátor (16 pozíciónként), illetve BASIC EDIT esetén az egész sor végigírása RETURN-el
10	A	CTL-J, Y	Outputként hatástalan, inputként BASIC EDIT ill. Linein esetén speciális SHIFT, a kódok értékéhez '40'-et ad hozzá
11	B	CTL-K	
12	C	CTL-L, CLS	Ernyőtörlés
13	D	CTL-M, RETURN	Új sor, BASIC EDIT esetén a maradék sor törlődik
14	E	CTL-N, CR	<del>Új</del> sor <i>eleje</i> (ugyanabban a sorban)
15	F	CTL-O	Függőleges írás. Az ernyő mérete ilyenkor 32*21, és a PRINT parancs koordinátái meg vannak cserélve.
16	10	CTL-P	Subscript - a következő kiírás 4 pontsorról lejjebb történik
17	11	CTL-Q	Superscript - a következő kiírás 4 pontsorról feljebb történik.
18	12	CTL-R	Duplaszéles char üzemmód kikapcsolás
19	13	CTL-S	Az ernyő alapszíne sötét
20	14	CTL-T	A karakter alapszíne az ernyővel egyező lesz
21	15	CTL-U	Aláhúzás kikapcsolás
22	16	CTL-V	Karakterpozíciók nem törlődnek kiírás előtt. (kompozit karakterek előállítása lehetséges)
23	17	CTL-W	Vízszintes írás (42*16)
24	18	CTL-X, SH <--	Linein funkciónál a beírt sor törlődik, BASIC EDIT esetén a sor editálása előlről kezdődik.
25	19	CTL-Y, -->	BASIC EDIT esetén a sor következő karaktere kiíródik az ernyőre



## BASIC hibajelzések

1	1	NF	NEXT hiba, NEXT használata megelőző FOR nélkül
2	2	SN	Szintaktikus hiba. Ha a sor visszairódik, rögtön javítható
3	3	RG	RETURN használata megelőző GOSUB nélkül
4	4	OD	Több adat olvasása READ utasítással, mint amennyi a DATA utasításokban meg van adva
5	5	FC	Függvényhívási (paraméterezési) hiba, vagy negatív tömbindex
6	6	OV	Túlcsordulás, az egyes ábrázolási pontosságok túllépése számolás közben
7	7	OM	Memóriahiány
8	8	UL	Definiálatlan címkére hivatkozás
9	9	BS	Indexhatár vagy dimenziószám túllépés
10	A	DD	Tömb újradimenzionálása. Leggyakoribb oka vagy több DIM utasítás használata, vagy a DIM utasítás nem előzi meg az első hivatkozást. (Az első hivatkozáskor a tömb minden indexe 0-10 határra deklarálódik)
11	E	/0	Nullával osztás
12	C	ID	Az utasítás parancsként nem használható
13	D	TM	Adattípus hiba
14	F	OS	String memória túlcsordulás. A hiba javításához a program elején meg kell adni a CLEAR nn utasítást, amely beállítja a String memória méretét. (alapértelmezés = 50)
15	F	LS	255 karakternél hosszabb string használata
16	10	ST	A megengedettnél bonyolultabb stringkifejezés használata
17	11	CN	A CONTINUE parancs illegális használata
18	12	NR	A RESUME parancs illegális használata
19	13	RW	A RESUME parancs érvénytelen
20	14	UE	Definiálatlan hiba
21	15	MO	Hiányzó operandus
22	16	FD	Hibás adat a file-ban, vagy korai filevég
23	17	L3	Floppydiszkes rendszer hibajelzése



# INDEX

<					szalagfelépítés	68
<-- gomb	12	12	24	24	terület	55
-					tipuskonverzió	28
					töltés assemblerből	84
					törlés	
					kézzel	23 26
--> gomb	24	24			törlés assemblerből	82
						83
A					utasításszeparátor	12
ABS	43				újraindítás	23 26
ALLOC	75				változók	(ld. változó)
ASC	41				végcim	61
Assembler program					BASIC RAM	
automatikus indulás	69				BASIC program	55
77					Belső változók	55
elhelyezés	21	53	61	61	fontosabb rekeszek	61
hivas BASIC-ből	53				program kezdet	61
hívható rutinok	70				skalár kezdőcim	61
konvenciók	54				STACK kezdőcim	61
paraméter BASIC-ből	53				STRING kezdőcim	61
regiszterhasználat	54				STRING kurrens pointer	61
relokálás	82				STRING végcim	61
STACK hossz	54				tömb kezdőcim	62
szalagfelépítés	68				tömb végcim	62
töltés assemblerből	84				helye	55
töltés BASIC-ből	21				skaláris változók	55
ATN	44				STACK	55
AUTO	20				STRING terület	55
					tömbök	55
B					BEEP	51 84
BASIC program					betöltés	
belső ábrázolás	19				assembler	
hibaellenőrzés	19				assemblerből	84
hibakeresés	26	27	29	30	BASIC-ből	21 54
hibakezelés	31	31	32	48	kézzel	54
48					BASIC	
indítás assemblerből	84				assemblerből	84
indítás kézzel	26				BASIC-ből	21
javítás	23				cim	69 76 81
kezdőcim					formátum	68
memóriában	61				vezérlés	58
szalagon	69	76			BRK gomb	12 20 23 24 25 25
kifejezések	28				56	70 71 71
kiiratás	24	25			C	
konstansok	16				CALL	53
leállítás	12	29	29		CDBL	44
műveletek					CHKSUM	80
jelei	17				CHRE	41
sorrendje	17				CINT	44
programláncolás	21				CLEAR	20 83
sorhossz	19				CLRS	82
sorszám	19	20				

CLS 53  
 CLS gcmb 12 20 24 71  
 CCMODEORE  
   ATTN 64 65  
   perifériák illesztése  
     11  
   Serial Clock 64 64  
   Serial Data 64 64  
   Serv. Req 60 64  
 CONT 23  
 COS 44  
 CREATE 40  
 CSNG 45  
 CTR gcmb 12  
  
**D**  
  
 DATA 38 39  
 DCB  
   +14-18 - nyomtató paramé-  
     terek 57 74  
   +20 - kazettakezelő opci-  
     6k 58 76 77 77  
     78 78 79  
   +26 - kazetta rekordtípus  
     58 76 76 76 77  
     77 77 79  
   +27 - kazetta hibaszámlá-  
     lő 58 79 81  
   +28 - kazetta rekordszám-  
     láló 58 76 77  
     77 81  
   +29,30 - kazettakezelő  
     belső változók  
     58 79  
   +3 display/printer opciók  
     56 74  
   +6 - képernyő X cím 57  
     70 72  
   +7 - képernyő Y cím 57  
     70 72  
   +8 - képernyő puffercím  
     57 75  
   +9,10 - karaktergenerátor  
     tábla címe 57  
     66  
   cím 54 55  
   tartalma 55  
 DCUES 72  
 DEF... 27  
 DELETE 23  
 DIM 28  
 DISPLAY 70  
 DROW (ld. grafika)  
 DRVOFF 79  
 DRVON 79

**E**  
  
 EDIT 23  
 ELSE 33  
 END 29  
 ERL 48  
 ERR 48  
 ERROR 31 32  
 EXP 45  
  
**F**  
  
 felfele nyíl gomb 17 17 36  
 file (ld. kazetta)  
 FIX 45  
 FOR 32  
 FRE 49  
 funkcionális billentyűk 12  
   24  
  
**G**  
  
 GOSUB 30 31  
 GOTO 29 30 31  
 grafika  
   DROW 56 56  
   POINT 73  
   RESET 73  
   SET 52 52 52 72  
  
**H**  
  
 hanggenerátor  
   felépítés 8  
   frekvencia 51  
   hangfelvétel 51  
   hívása assemblerből 84  
   hívása BASIC-ből 51  
   időtartam 51  
   kazettakezelésnél 58 76  
     77 77 78 78 78  
 hibakezelés  
   BASIC programban 31 31  
     32 48 48  
  
**I**  
  
 IF 33  
 INBYTE 80  
 indítás  
   assembler  
     automatikus 54 69  
     BASIC-ből 54  
     kézzel 54  
   BASIC  
     assemblerből 84  
     kézzel 54  
 INKEY 38 70



INP	49	olvasás várakozással	
INPUT	37 71	71	
INPUT#	40	sor olvasása	71
INT	45	kezelő flag-ek	56 56
IT	59 64 64	mentés	68
		mérete karakterben	7 65
J		mérete pontban	7 51 65
		puffer	
Joystick		allokálás	75
felépítés	10	helye	51 55 65
kezelés	63 65 65	mérete	7 51 55 57 57
			65 72 73
K		váltás	75
		puffercim	57 63
karaktergenerátor		törlés	82
karakterkép	66	üzenet	
tábla felépítés	66	szalagkezelés	58 79
táblacim	57		80
vezérlés	66	vezérlő karakterek	7 35
kazetta			88
adathlokk	68 76 81	klaviatúra	
BASIC program kezdőcim	61	felépítés	6
BASIC program végcim	61	kezelés	(ld. még képer-nyő) 64
filenév	68 76 79	konstansok	16
hangfelvétel	51		
header	68 76 79	L	
hibaszámláló	58 79 80		
	81	lefele nyil gomb	19 24 56
illesztő input	60 64		71
illesztő output	63	LEFT	41
interface	9	LEN	41
kontrolösszeg	68 69 69	LET	28
	78 78 80 80 81	LINEIN	71
opciók	58 76 77 77 78	LIST	24
	78 79	LLIST	25
program kezdőcim	69 77	LN	46
rekordsorszám	79 80 81	LOAD	21 (ld. betöltés) 84
rekordszámláló	58 68 68	LOG	46
	69 76 77 77 77	LPRINT	37
rekordtípus	58 68 68 69	LPRINT USING	37
	76 76 77 79 80		
távvezérlés	63 63 76 77	M	
	79 79 79		
trailer	69 77	magno	(ld. kazetta)
képernyő		Megszakítás	59 64 64
címzés	57 57 72 74	MIDE	42
fizikai input	64 64		
frissítés	7 64 75	N	
grafika	(ld. grafika)		
illesztés	7	NEW	26 83
karakterkészlet	7 87	NEWC	82
karakterkészlet módosítás	8 66	NEXT	33
kezelés	70	NMI - non-maskable IT	
kiíratás	70	engedélyezés/letiltás	63
olvasás várakozás nél-		REAL-TIME óra	59
kül	70	RESET	60



nyomtató (ld. printer)

## O

ON 30  
OPEN 39  
OUT 49

## P

PEEK 50  
POINT 52 73  
(ld. grafika) 73  
POKE 50  
POS 49 74  
PRINT 12 34 74  
PRINT USING 35  
PRINTR 35  
PRINT# 40  
printer  
illesztés 9  
karakterkészlet 74  
karakterkivitel 74  
paraméterek 57  
vezérlőkódok 74  
1F, 1F konverzió 56 74  
processzor  
seleesség 5  
típus 5  
PWRRESET 81

## R

RANDM 46 47 61  
RDDATA 81  
REAL 38 39  
REAL-TIME óra  
cime 59  
kikapcsolása 60 63  
növeleése 59  
RELOC 82  
REM 12 34  
RESET 52 73  
(ld. grafika) 73  
bekapcsolási 81  
gomb 6 12 60 63 64  
RESFT gomb 25  
RESTORE 39  
RESUME 31  
RETURN 30  
RETURN gomb 12 20 24 37 71  
RIGHT 42  
RND 46 47 61  
RST  
cimek és funkciók 59  
szabadon használható 59  
RUN 26 84 (ld. indítás)

## S

SAVE 22 85  
SET 52 72  
(ld. grafika) 72  
SGN 47  
SHIFT gomb 12 12 24 24 71  
SIN 47  
skaláris változó 55  
formája 15  
helyének keresése 83  
kezdőcim 61  
speciális SHIFT 56 60  
SQR 47  
SRCHDR 79  
SRCSYN 80  
STACK  
áthelyezés 61 82 83  
helye 55  
hossz 54  
inicializálás 60  
kezdőcime 61  
töltés RUN-hoz 84  
STEP 32  
STOP 29  
STRE 43  
STRINGE 42  
SWITCH 75  
system RAM  
helye 55  
inicializálás 60  
kiosztás 55 59  
szalag (ld. kazetta)

## T

TAB 35  
TAN 48  
TEST 22 85  
THEN 33  
TO 32  
tömbváltozó  
helye 55  
kezdőcime 62  
leírása 15  
végcime 62  
TROFF 27  
TRON 26

## U

UPPER gomb 12 56

## V

VAL 43  
VARPTR 50 83

változó	
double	
ábrázolás	14
egész	
ábrázolás	13
logikai	
ábrázolás	13 17 33
név	13 15
single	
ábrázolás	14
skaláris	
allokálás	83
hely	55
keresés	83
kezdőcim	61
leírás	15
string	
ábrázolás	14
típusdeklaráció	13 27
tömb	
ábrázolás	15
dimenzió	15 15 28
hely	55
kezdőcim	62
leírás	15
végcím	62
törlés	21
V24 interface	10 63 63 64

## W

WRBYTC	78
WRBYTE	78
WRDATA	76
WRHDR	76
WRSYN	77
WRTBL	77